uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2017-2018

Trabajo Fin de Grado

"Análisis, diseño e implementación de una solución de monitorización de Aula Informática basada en ELK"

Rafael Mateos López

Tutor

Alejandro Calderón Mateos

Leganés, Julio 2018

RESUMEN

El actual Trabajo Fin de Grado presenta un sistema de monitorización en tiempo real gratuito que tiene como base el conjunto de componentes de la empresa Elastic, el *ELK Stack* (Elasticsearch, Logstash, Kibana). Además, cada una de las herramientas que compone este sistema se desplegará mediante contenedores de software o de aplicación, los cuáles son fácilmente configurables a través de la herramienta Docker.

Este sistema desarrollado se encarga del análisis de *logs* de sistemas operativos como Linux con el objetivo de poder resumir, reunir y representar gráficamente la información relevante de dichos archivos y así poder facilitar su entendimiento y monitorización a un usuario administrador.

La monitorización obtenida a través del sistema puede ser útil, por ejemplo, para reunir información acerca del rendimiento del ordenador (ocupación de memoria, uso de CPU, temperatura del procesador) o para conocer los distintos intentos de inicio de sesión que se realizan en un determinado equipo, lo que puede llegar a descubrir un ataque de adivinación de contraseña por fuerza bruta.

Además, el sistema ha sido creado específicamente para monitorizar los ordenadores de las aulas informáticas de la universidad, permitiendo así que un usuario administrador como, por ejemplo, un miembro del Departamento de Informática, pueda controlar diferentes aspectos de todos los equipos de un aula desde un único ordenador en el departamento. Por otro lado, esto no impide que el sistema no pueda usarse en un entorno con un número menor de ordenadores donde interese su aplicación.

La elección del sistema base, así como de cada una de las herramientas que forman el sistema, ha sido realizada tras la comparación entre las opciones más conocidas de cada uno de los procesos incluidos en el análisis de logs, creando así un sistema que permita realizar este análisis y, por ende, la monitorización objetivo, de la forma más completa posible.

Palabras clave: logs, monitorización, contenedores de software, Docker, Logstash, Elasticsearch, Kibana.

Debido a que este trabajo se corresponde con un Trabajo Fin de Grado del Grado Universitario en Ingeniería Informática del Plan 2011, se exige un resumen en inglés de mínimo 10 páginas para demostrar la competencia en dicho idioma. Por ello, este resumen se encontrará en el siguiente apartado del actual documento.

ABSTRACT

INTRODUCTION

In the last decades, a great evolution has taken place in the computers. This evolution is reflected in both the improvement of the computers physical components and the development and improvement of new security systems to guarantee the computer and the user information security. However, at the same time this security improvement was taken place, the number of threads and developed malware was also rising and their quality and complexity was getting bigger in comparation to the one they had some years ago.

These new computer security threads don't put at risk only the private information of the users, but they also, in some cases, can produce unwanted behaviours in the physical components of the computer (an increase of his temperature, an increase of his fans speed, etcetera) what can prevent its normal performance.

Moreover, in environments with a big number of computers, the process to control the security and the normal behaviour of each computer can be an expensive process (referred to the time spent to do it). This is the case of each computer classroom in the university, where its normal to see around 15-20 computers in each classroom.

This is the context where the idea of this project comes up: to monitor the logs information from some computers with a Linux operative system to collect, summarize and control the relevant information inside those files through a group of visualizations, easier to handle and understand by users. This is the main purpose of the **log analysis**.

Apart from that, in the last years a new virtualization environment has come up as a simpler and cheaper solution that virtual machines: these are the **software containers**. These containers have a great advantage: they don't require the installation or configuration of the entire operative system, but they use the operative system of the system they use as an base image, and they incorporate just enough services and libraries to make possible the execution of the applications that are going to be installed inside the container.

The idea of this last paragraph is the reason why every program used in this work is going to be deployed using software containers, taking advantage of their advantages over the local installation in the operative system and the virtualization with virtual machines.

OBJECTIVES

The main objective of this work is the creation and configuration of a free system that allows the extraction of the computers log information, its storage in a database or any other storage system and its later graphic representation to be monitored in real time. These computers whose information is extracted will have a Debian operative system to simulate the operative system installed in the computers of some of the computer classrooms of the university and allow its later incorporation and deployment in those computers.

Moreover, the second objective of this work is to develop a system that has some security mechanisms to encrypt the intercommunication of every tool that is part of the system and the storage system where the information received from each computer Is storage.

STATE OF THE ART

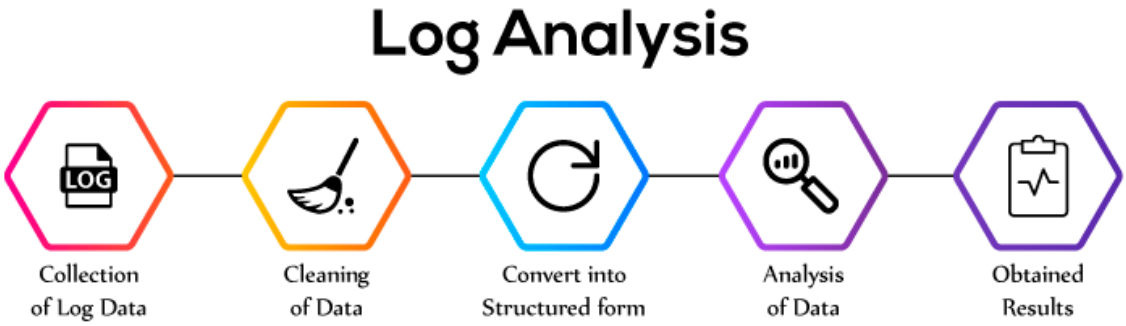
The log analysis is a technique that has been developed in the last years with the main purpose of reducing the difficulty of understanding the information inside these types of logs and summarize that information to extract the one that can be more important to the user, at first instance.

This technique is especially useful when some errors occur in the system since the administrator or developer of this system can find the problem origin in an easier and faster way and, from there, trigger some answers to solve the problem.

Furthermore, the fact that most part of the computers relevant information is present in the system as log files, as well as the different events that occurs in these computers and in the programmes installed in them, makes the interest of development of this technique much bigger.

As a result, there are a lot of tools that have come up to make this analysis in a more complete and efficient way. This, a complete analysis log system is the one that offers the next functionalities:

- Read each of the lines logs, remove the redundant information and filter the relevant information to get its conversion in fields or data structures.
- Save the logs read information in a data storage (like databases, clusters, etcetera).
- Access to this data storage and offer a visualization and a real-time monitorization from the stored information in a different way than the used in the log files.



In this way, the existent tools for log analysis can be classified in those that are based in one or two of the functionalities before mentioned (not the three at the same time) and, therefore, are often used alongside other tools or programmes to offer to users a more complete analysis, and those that includes the three functionalities inside the same product, not being necessary its deployment alongside others programmes to extend its functionalities.

The solution presented in this work is in this first mentioned group: the components of the ELK Stack (Elasticsearch, Logstash and Kibana) that, although it consists in three different products, they all belong to the same company, Elastic. On the other side, in the second mentioned group, the ones that includes all the functionalities to work by themselves, the most well-known tools are Sumo Logic and Splunk.

Comparison between ELK Stack, Sumo Logic and Splunk

In the next table, it can be found a comparison between some aspects of the free licenses of the tools above mentioned.

	Sumo Logic FREE	Splunk FREE	ELK Stack (Basic)
Data collection and forwarding	✓	✓	✓
Data storage	✓	✓	✓
Searching	✓	✓	✓
Analysis	✓	✓	✓
Visualization	✓	✓	✓

	Sumo Logic FREE	Splunk FREE	ELK Stack (Basic)
Cloud service	✓	✗	✗
Limitations	500 MB/day	500 MB/day	Unlimited
Plugins	✓	✓	✓
Integrations	✓	✓	✓
Security	✓	✗	✓
Control access	✓	✗	✓
Documentation and tutorials	✓	✓	✓
Community and forums	Read-only	Read-only	Read and limited creation of posts
Support	✗	✗	✗

Each of the three products has far better features when they are installed using payment licenses. In this situation where every product has its more expensive license running, Splunk would be the one that gives more and better features to the user. However, the cost of the license and the maintenance of the products with those licenses would make them practically untenable in a not very wealthy company.

In the case of this work, what is being looking for is a solution that allows the log analysis and management in the most complete way, but also for free, so the cost of the product doesn't become a reason that prevents its incorporation to the university community.

The idea of the last paragraph, the offering of almost all the features in the table above and its little limitations in its free version, are the reasons why the solution chosen to deploy as a system base in this work is the **ELK Stack**. Furthermore, although one the biggest problems of this solution is the effort needed to its deployment and configurations because of the quantity of tools that have to be configured, at the end of the actual document there will be some manuals to make that configuration and deployment an easier process.

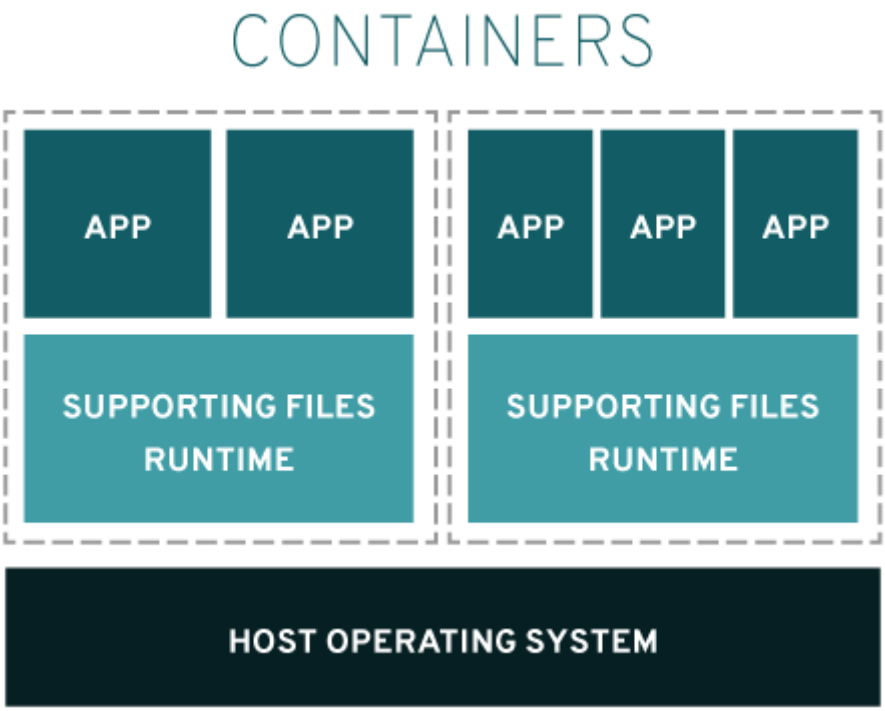
DESIGN

Tools selection

Local installation VS Docker Containers

At a first instance, every tool that was part of the developed system was going to be installed locally in the operative system. However, in the last years a new way of virtualization has come up: this is the system operative virtualization or software container based virtualization.

But, what are the software containers? The software containers are environments with a low time of execution which provide the necessary variables and libraries needed by the applications to be executed, maximizing their portability this way.



Some of the main features of the software containers are the following ones:

- **Easy installation**: containers are installed through images, what could be defined as little operative systems with a few applications installed. Once the user assign the desired configuration to the image, the deployment of the container can be resumed in the execution of just one command.
- **Great portability**: once the container is configured, it can be easily transferred to any other systems thanks to its independency from the platform it is deployed in. The only requirement to make this possible is that the end system has to support this type of technology.
- **Isolated applications**: installed applications in a container are completely independent from the ones installed in other containers. This way, applications with different even opposed requirements, could work correctly in the same system.

- **Self-sufficiency**: following the definition previously given, one of the advantages of the containers is that they are built with just the needed files and libraries to make possible the execution of the applications that are installed in the container.

Because of these last features, which are advantages against the local installation in the operative system, it has been decided to deploy each of the tools of the deployed system using software containers. Moreover, it has been decided the use of the tool **Docker** to control these containers as it makes easier its creation and deployment.

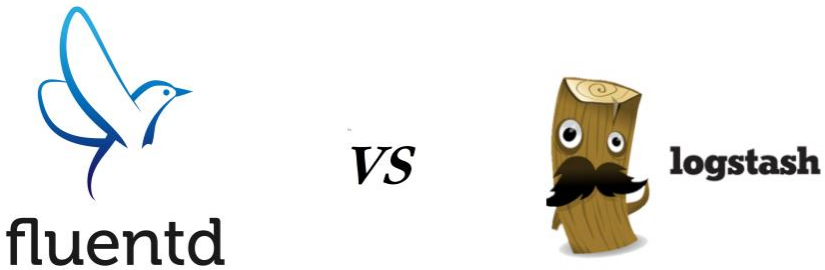
Solr VS Elasticsearch

Apache Solr and Elasticsearch are the main search engines used in the log analysis and both do their work in a correct but different way.

On the one hand, Solr is characterized by its support to a large amount of input and ouput formats, a rich documentation and a faster search speed when static data are the type of data stored. On the other hand, Elasticsearch offers a larger library support, an autonomic cluster, an easier process of learning and installation and a faster search speed of dynamic data.

Although each of the tools has its own advantages, Elasticsearch has been chosen to form part of the system because of its easy learning and installation and its autonomic cluster. Furthermore, one of the reasons of choosing **Elasticsearch** is that it's a component of the ELK Stack, which brings a great amount of useful tool to make a complete log analysis system.

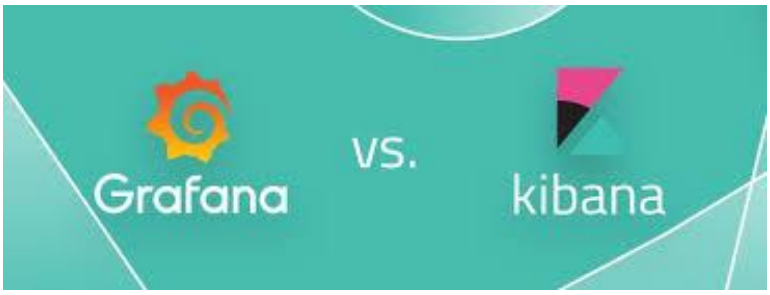
Fluentd VS Logstash



In this case, Fluentd and Logstash are known because of their great work in the management of logs. They both manage logs in a different way, they are executed in the most common platform and need other tools to transfer data to bigger systems.

As any of the tools could be chosen to be incorporated in the developed system because of their similitudes, **Logstash** is the final tool chosen because of its membership to the ELK Stack.

Grafana VS Kibana



Data visualizers are used to create graphic representation of the data that is stored in databases, clusters or other types of data storages. The most well-known data visualizers are Grafana and Kibana and one of them is going to form part of the system.

In this case, the selection of the final tool is much easier than in the last comparisons because Kibana is the only tool that offers log analysis as a included feature. The purpose of Grafana is the metrics analysis, it has a bigger number of possible integrations (in comparation to Kibana) and it offers a control access system by default. But the purpose of Grafana isn't the one that is being looked for.

So, in the end, **Kibana** is the data visualizer chosen to farm part of the system. Like Logstash and Elasticsearch, Kibana is a component of the ELK Stack, so its integrations with the other components of the stack won't be as harder as if any external tool would have been chosen.

Logstash and/or Filebeat?

Although Logstash was considered at first instance as the data forwarder of the system, it has a big problem: when it forwards the data of many computers, many pipelines have to be created to manage each connection and each of these pipelines consume a lot of memory in its creations. That's what makes Logstash untenable when it comes to the data forwarding of many computers at the same time.

That is the origin of the data forwarders like a Lumberjack or Logstash Forwarder. But some years ago, Elastic developed a group of tools to forward different types of data: the Elastic Beats. The tool of this group that is interested to be part of the system is Filebeat, which oversees the data forwarding from files (that's why it is called that way).

But, why should Filebeat be installed with Logstash instead of without it? As it was mentioned before, Logstash needs Filebeat to reduce its memory consumption, but it is also true that Filebeat needs the powerful filter of Logstash because it doesn't have any way of parsing or transforming the data collected.

Because of the reasons described above, Filebeat was included to the system as the data forwarder of the system , while Logstash would just parse and transform the data forwarded.

X-Pack and/or Search Guard?

X-Pack is plugin that can be installed in the ELK Stack to extend its functionalities with services as monitoring, alerting or security. Although this plugin is known as the best component to enable security mechanisms in the system with an SSL/TLS encryption and a powerful role-based access control, one of its biggest problems is that a payment license must be owned to enjoy these features.

There is where Search Guard comes up as a free solution to improve the security of the system. It offers the same features than X-Pack (in a simpler version) but it has the advantage that all of the main features are included in the free license.

That's why **Search Guard** became the next tool to be part of the developed system, with the purpose of enhancing its security and encrypt its communications.

However, X-Pack monitoring is some of the few features that are included in the basic license of Elasticsearch and this service can be useful to monitor the monitoring system itself, what is an interesting way of getting rid of the limitations of monitoring of the system, going one step ahead. This is the reason why **X-Pack** was also chosen as a tool of the developed system.

Arquitecture Design

After mentioning each tool involved in the system, now its turn to talk about the design of the system architecture.

To create the desired monitorization system its necessary the distinction of two type of computers, based on its function in the log analysis process:

- **Client(s):** one or more computers whose logs are processed to be monitored by the server.
- **Server**: the computer where the monitorization of the clients takes place.

Following this division, each type of computer will have a different group of tools installed. As an exceptional case, both clients and server will have installed Docker to deploy each tool in a software container.

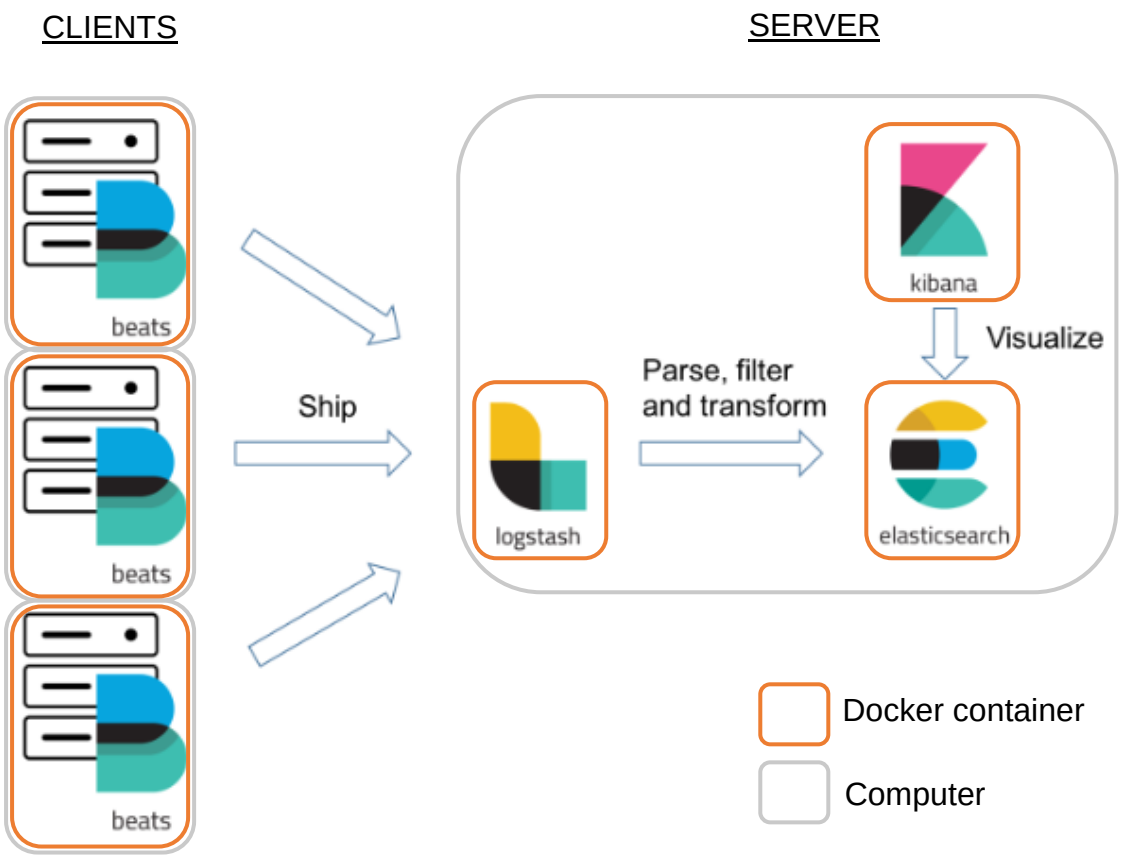
That being said, in the clients the only tool that is installed is Filebeat, because its data forwarding is the only thing needed in the client part of the system.

On the other hand, all the other tools must be installed in the server. This includes Logstash, Elasticsearch and Kibana as the main programmes, and X-Pack and Search Guard as plugins.

Thus, a summary of the analysis process that occurs in the system is the described in the next steps:

- In the client, Filebeat creates a harvester that reads and forwards the information of the logs to the server.
- Logstash receives the information, applies a group of filters to parse and transform the received data and injects the result in Elasticsearch, which stores the information in nodes and indexes to its later search.
- Kibana access to the Elasticsearch indexes and obtains the stored data to show it as graphs and dashboards.

The following image is a visualization of how the analysis log process works and how the system architecture is finally configured.



CONCLUSIONS

First, it must be said that the main objective has been accomplished: the creation of a system able to monitor the log information from different computers through graphics and dashboards in real time. It has also been accomplished the creation of the desired and detailed documentation, to make easier the configuration and deployment of each tool involved in the system.

However, it wasn't possible to enable the desired security mechanisms because of the many errors that occurred in the validation of certificates from different tools. Anyways, the role-based control access and the encryption of the Elasticsearch cluster could be added, so it's not like any security service was implemented.

Secondly, although many attended subjects of the career helped in some of the aspects of the work, the realization of this work involved the learning and configuration of a big amount of tool tools, which weren't known until the start date of the project. This added a lot of difficulty to the creation and deployment of the system and time to get these processes done.

Finally, as future works, the main thing that needs to be done to offer a fully secured system is to complete the certificate configuration offered by Search Guard, which gave a lot of problems in the actual work and was impossible to get it done before the deadline.

The other main future work would be to create more uses to the developed system. This work is a functional system to monitor the information of any desired log file, so it's only needed to define which logs could be monitored. One of the main uses that is going to be implemented in the future is the injection into logs of information related to the computers performance (memory consumption, processor temperature, etcetera) and its later monitoring through the developed system. This way, it could be possible to detect any anomalies in the normal behaviour of the systems just looking at some graphics or dashboards in the screen.

AGRADECIMIENTOS

Quería agradecer y dedicar este trabajo, en primer lugar, a mis padres y familiares, por haberme apoyado desde el primer momento que decidí empezar y completar este grado.

A los distintos compañeros de clase que he tenido durante estos 4 años, por hacer de las asignaturas y sus correspondientes clases un entorno basado en la ayuda y la colaboración, en vez de una lucha por el éxito individual.

A mis amigos con los que he compartido la carrera, Mario, Volodymyr y Víctor, por haber cooperado conmigo para reducir en gran medida el esfuerzo necesario para completar las prácticas y aprobar las asignaturas, a la vez que hacer los intensos días de estudio mucho más llevaderos y menos estresantes.

A los profesores que me han impartido clase, por haberme enseñado de forma profesional el contenido de sus asignaturas y haberme ayudado a conocer qué ámbitos de la informática me interesan por encima del resto.

A Alejandro Calderón, por haberme animado desde el momento en el que lo elegí como tutor de mi TFG, por haberme ayudado y aclarado mis dudas cuando no sabía cuál debía ser mi siguiente paso y, sobre todo, por haber confiado en mí desde el primer momento.

¡Muchas gracias a todos!

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN 1

1.1. MOTIVACIÓN 1
1.2. OBJETIVOS 2
1.3. ALCANCE 2
1.4. ESTRUCTURA DEL DOCUMENTO 3

2. ESTADO DE LA CUESTIÓN 5

2.1. INTRODUCCIÓN 5
2.2. SUMO LOGIC 6
2.3. SPLUNK 7
2.4. COMPARACIÓN DE LAS DISTINTAS SOLUCIONES Y LA PROPUESTA 7

3. INTRODUCCIÓN AL DISEÑO, ANÁLISIS, IMPLEMENTACIÓN E IMPLANTACIÓN 11

3.1. METODOLOGÍA USADA 11
3.2. CICLO DE VIDA 11
3.3. MARCO REGULADOR 11

4. ANÁLISIS 13

4.1. REQUISITOS 13
4.1.1. Requisitos funcionales 14
4.1.1.1. Requisitos funcionales del equipo cliente 14
4.1.1.2. Requisitos funcionales del equipo servidor 16
4.1.2. Requisitos no funcionales 19
4.1.2.1. Requisitos no funcionales del equipo cliente 19
4.1.2.2. Requisitos no funcionales del equipo servidor 20
4.1.2.3. Requisitos no funcionales comunes 22

5. DISEÑO 23

5.1. SELECCIÓN DE LAS HERRAMIENTAS 23
5.1.1. Instalación local VS Contenedores Docker 23
5.1.2. Solr VS Elasticsearch 24
5.1.3. Fluentd VS Logstash 26
5.1.4. Grafana VS Kibana 27
5.1.5. ¿Logstash y/o Filebeat? 28
5.1.6. ¿X-Pack y/o Search Guard? 29
5.2. DISEÑO DE LA PROPUESTA 30
5.2.1. Docker 30
5.2.2. Elasticsearch 31
5.2.3. Logstash 31
5.2.3.1. Filtro grok 32
5.2.4. Kibana 33
5.2.4.1. Parámetros para la visualización 33

5.2.4.2. Consola de comandos *34*
5.2.4.3. Tipos de visualizaciones *35*
5.2.5. Filebeat 36
5.2.6. X-Pack 37
5.2.6.1. Monitorización *37*
5.2.6.2. Search Profiler *38*
5.2.6.3. Grok Debugger *39*
5.2.6.4. Control de acceso basado en roles *39*
5.2.7. Search Guard 40
5.2.8. Arquitectura del sistema desarrollado 41
5.3. CASOS DE USO 43
5.3.1. Matriz de trazabilidad 48
6. IMPLEMENTACIÓN E IMPLANTACIÓN 49
6.1. TECNOLOGÍAS NECESARIAS 49
6.2. DESPLIEGUE DEL SISTEMA 50
6.2.1. Instalación y configuración de las herramientas 50
7. EVALUACIÓN 53
7.1. CASOS DE PRUEBA 53
7.1.1. Matriz de trazabilidad 57
8. PLANIFICACIÓN Y PRESUPUESTO 59
8.1. PLANIFICACIÓN 59
8.1.1. Planificación estimada 59
8.1.1.1. Diagrama de Gantt *63*
8.1.2. Planificación real 63
8.1.2.1. Diagrama de Gantt *65*
8.2. PRESUPUESTO 66
8.2.1. Gasto de personal 66
8.2.2. Gastos de material 67
8.2.3. Otros gastos directos 68
8.2.4. Presupuesto final 71
9. CONCLUSIONES Y TRABAJOS FUTUROS 73
9.1. CONCLUSIONES 73
9.1.1. Producto 73
9.1.2. Proceso 73
9.1.3. Personales 74
9.2. TRABAJOS FUTUROS 75
APÉNDICE I: MANUAL DE INSTALACIÓN DE DOCKER Y DOCKER COMPOSE 77
APÉNDICE II: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE LOGSTASH, ELASTICSEARCH Y KIBANA 79

APÉNDICE III: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE FILEBEAT. CONEXIÓN ENTRE FILEBEAT Y LOGSTASH.....81

APÉNDICE IV: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE X-PACK.....84

APÉNDICE V: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE SEARCH GUARD.....85

APÉNDICE VI: MANUAL DE OBTENCIÓN E INSTALACIÓN DE UNA LICENCIA TIPO BASIC EN ELASTIC 89

BIBLIOGRAFÍA.....91

ÍNDICE DE ILUSTRACIONES

Ilustración 1. Proceso del análisis de datos 5
Ilustración 2. Logo de Sumo Logic 6
Ilustración 3. Logo de Splunk 7
Ilustración 4. Ciclo de vida del proyecto 11
Ilustración 5. Virtualización basada en contenedores de software 23
Ilustración 6. Logo de Docker 30
Ilustración 7. Logo de Elasticsearch 31
Ilustración 8. Logo de Logstash 31
Ilustración 9. Ejemplo del filtro grok en Logstash 32
Ilustración 10. Logo de Kibana 33
Ilustración 11. Interfaz gráfica de Kibana 33
Ilustración 12. Actualización automática de Kibana 34
Ilustración 13. Rango de tiempo en Kibana 34
Ilustración 14. Consola de comandos de Kibana 35
Ilustración 15. Tipos de visualizaciones de Kibana (I) 35
Ilustración 16. Tipos de visualizaciones de Kibana (II) 36
Ilustración 17. Logo de Beats 36
Ilustración 18. Logo de X-Pack 37
Ilustración 19. Monitorización de X-Pack en Kibana 38
Ilustración 20. Search Profiler en Kibana 38
Ilustración 21. Grok debugger 39
Ilustración 22. Control de acceso basado en roles 39
Ilustración 23. Logo de Search Guard 40
Ilustración 24. Inicio de sesión para acceder a Kibana 40
Ilustración 25. Arquitectura del sistema 42

ÍNDICE DE TABLAS

Tabla 1. Comparación entre Sumo Logic, Splunk y ELK Stack 8
Tabla 2. Plantilla de los requisitos 13
Tabla 3. RF-01 14
Tabla 4. RF-02 15
Tabla 5. RF-03 15
Tabla 6. RF-04 16
Tabla 7. RF-05 16
Tabla 8. RF-06 17
Tabla 9. RF-07 17
Tabla 10. RF-08 18
Tabla 11. RF-09 18
Tabla 12. RF-10 19
Tabla 13. RNF-01 19
Tabla 14. RNF-02 20
Tabla 15. RNF-03 20
Tabla 16. RNF-04 21
Tabla 17. RNF-05 21
Tabla 18. RNF-06 22
Tabla 19. Comparación entre Solr y Elasticsearch 25
Tabla 20. Comparación entre Fluentd y Logstash 26
Tabla 21. Comparación entre Grafana y Kibana 27
Tabla 22. Herramientas en cada tipo de equipo 41
Tabla 23. Plantilla de los casos de uso 43
Tabla 24. CU-01 45
Tabla 25. CU-02 45
Tabla 26. CU-03 46
Tabla 27. CU-04 47
Tabla 28. CU-05 48
Tabla 29. Matriz de trazabilidad requisitos - casos de uso 48
Tabla 30. Plantilla de los casos de prueba 53
Tabla 31. PRU-01 54

Tabla 32. PRU-02 54
Tabla 33. PRU-03 55
Tabla 34. PRU-04 56
Tabla 35. PRU-05 56
Tabla 36. Matriz de trazabilidad casos de uso - casos de prueba 57
Tabla 37. Planificación estimada (esfuerzos) 61
Tabla 38. Planificación estimada 62
Tabla 39. Diagrama de Gantt estimado 1 63
Tabla 40. Diagrama de Gantt estimado 2 63
Tabla 41. Diagrama de Gantt estimado 3 63
Tabla 42. Planificación real 65
Tabla 43. Diagrama de Gantt real 1 65
Tabla 44. Diagrama de Gantt real 2 66
Tabla 45. Diagrama de Gantt real 3 66
Tabla 46. Gastos de personal 67
Tabla 47. Gastos de material 68
Tabla 48. Gasto de la luz 69
Tabla 49. Gasto de ADSL 69
Tabla 50. Otros gastos indirectos 70
Tabla 51. Información de gastos indirectos 70
Tabla 52. Presupuesto final 71

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

En las últimas décadas, la evolución en el campo de la informática ha sido bastante notable. Esta evolución se ve reflejada tanto en la mejora de los componentes físicos de los equipos informáticos (procesadores, tarjetas gráficas, placas base, etc.) como en el desarrollo y mejora de nuevos sistemas para garantizar su seguridad y la de la información del usuario. Sin embargo, junto a esta mejora de la seguridad informática, también ha crecido el número de amenazas y *malware* desarrollado, de la misma forma que lo ha hecho su calidad y complejidad en comparación con la que les caracterizaba hace, simplemente, unos pocos años.

Estas nuevas amenazas de la seguridad informática no sólo ponen en riesgo la información privada de los usuarios, sino que, en algunos casos, también pueden generar comportamientos indeseados en los componentes físicos del ordenador (aumento de la temperatura o de la velocidad de los ventiladores, etc.) lo que puede llegar a impedir su normal funcionamiento.

Además, en entornos que cuenten con un gran número de ordenadores, el proceso de controlar la seguridad y el normal comportamiento de cada uno de los equipos puede ser un proceso costoso, en lo referido al tiempo que se necesitaría para ello. Este es el caso de las aulas informáticas de la universidad, donde son usuales unos 15-20 ordenadores por aula.

Es en este contexto donde surge la idea presente en este trabajo: monitorizar la información contenida en los *logs* de varios equipos que cuenten con sistemas operativos como Linux con el objetivo de poder reunir, resumir y controlar la información relevante de estos archivos a través de un conjunto de gráficos, más fáciles de manejar y de entender por los usuarios. Este es el principal objetivo del **análisis de *logs***.

Por otro lado, en los últimos años ha surgido un nuevo entorno de virtualización mucho más simple y barato (en cuestión de recursos) que las máquinas virtuales: los **contenedores de aplicaciones**. Estos contenedores tienen una ventaja principal: no requieren la instalación y configuración de un sistema operativo completo, sino que utilizan el sistema operativo del sistema de una imagen base, e incorporan únicamente una serie de servicios y librerías mínimos para la ejecución de aplicaciones.

Es por esta razón por la que el sistema presentado en este trabajo se desplegará utilizando contenedores de aplicaciones, aprovechando así sus ventajas frente a la instalación en el propio sistema operativo y la virtualización mediante máquinas virtuales.

Por último, cabe destacarse que en la actualidad son muchas las herramientas que existen para cada una de las actividades incluidas en el proceso de análisis de *logs* (recolección de información, almacenamiento, visualización, etc.). Cada una de estas herramientas tienen diferentes versiones y archivos de configuración cuyo formato va cambiando constantemente.

De esta forma, aunque es cierto que existen tutoriales para la instalación y configuración de cada una de estas herramientas, la dificultad de configuración e interconexión entre estos elementos es mucho mayor cuando las versiones de cada uno de ellos difieren o cuando el número de elementos que forman parte del sistema no se resume únicamente en un par de ellos.

1.2. OBJETIVOS

El principal objetivo de este trabajo es la creación y configuración de un sistema gratuito que permita la extracción de información de los *logs* de equipos informáticos, su almacenamiento en una base de datos u otro sistema de almacenamiento y su posterior representación gráfica para ser monitorizados en tiempo real. Estos equipos informáticos de los que extrae la información contarán con un sistema operativo Debian, simulando el presente en los equipos de las aulas informáticas del Departamento de Informática de la Universidad Carlos III de Madrid y permitiendo así su posterior incorporación y despliegue en dichos equipos.

Además, como segundo objetivo se buscará que el sistema desarrollado presente una serie de mecanismos de seguridad que permitan cifrar las comunicaciones entre las distintas herramientas que lo forman, así como el sistema de almacenamiento donde se vaya incorporando la información recibida de cada uno de los ordenadores.

Por último, se buscará la creación de una documentación completada y detallada que permite la creación y simulación del sistema, facilitando todo el proceso de instalación, configuración y despliegue de cada una de las herramientas incorporadas, ya que no son usuales los tutoriales que incluyen la interconexión de tantas herramientas con detalle.

1.3. ALCANCE

Para conseguir los objetivos anteriores ha sido necesario la realización de las siguientes tareas:

1) Análisis del estado actual del análisis de *logs* junto al análisis de las principales herramientas utilizadas para su realización.
2) Elección y aprendizaje del sistema base y de las distintas herramientas que lo formarán/complementarán.

3) Instalación y configuración de cada una de estas herramientas.
4) Establecimiento de un sistema de seguridad que cifre las comunicaciones entre las herramientas que forman el sistema.

1.4. ESTRUCTURA DEL DOCUMENTO

Antes de indicar los distintos capítulos en los que se divide el documento, se indicará una serie de reglas de formato que se seguirán a lo largo del documento:

- Los nombres en inglés, a excepción de nombres propios, aparecerán *en cursiva*. Por ejemplo: *log*.
- Los nombres de archivos o rutas de ficheros aparecerán "*en cursiva y entrecomillados*". Por ejemplo: *"/path/to/logs*".

Con respecto a la estructura del actual documento este se encuentra dividido en los siguientes capítulos:

- **Introducción**: capítulo que incluye los aspectos generales del trabajo (motivación, objetivos, alcance). Coincide con el capítulo actual del documento.
- **Estado de la cuestión**: capítulo donde se compara la solución propuesta en este trabajo con otras ya existentes que resuelvan el problema, aunque en menor medida o con un mayor número limitaciones.
- **Introducción al análisis, diseño, implementación e implementación**: capítulo donde se exponen las ideas generales del proceso de desarrollo del sistema (metodología usada, ciclo de vida del y marco regulador).
- **Análisis:** capítulo en el que se presentan tanto los requisitos funcionales como los no funcionales del sistema.
- **Diseño:** capítulo donde se muestra la arquitectura general del sistema, indicando las herramientas que lo forman tras compararlas con otras posibles opciones. Además, se desarrollan los casos de uso del sistema.
- **Implementación e implantación:** capítulo donde se indican las versiones de los programas o herramientas usados en el sistema, así como el proceso general de despliegue del sistema.
- **Evaluación:** capítulo donde se comprueba el correcto funcionamiento del sistema, indicando para ello los distintos casos de prueba y realizando pruebas utilizando el sistema desarrollado.
- **Planificación y presupuesto:** capítulo en el que se presentan las planificaciones estimada y real del proyecto, acompañadas de un diagrama de Gantt, junto al presupuesto que lleva asociado el desarrollo del proyecto.

- **Conclusiones y trabajos futuros:** capítulo que incluye las conclusiones personales y conclusiones del sistema desarrollado y del proceso que ha llevado a su desarrollo. También, en este apartado, se discuten los posibles o necesarios trabajos futuros que den lugar a una mejora del sistema o a la cobertura de alguna de sus actuales carencias.

2. ESTADO DE LA CUESTIÓN

2.1. INTRODUCCIÓN

El análisis de *logs* es una técnica que se desarrollado en los últimos años con el principal objetivo de reducir la dificultad para entender la información contenida en este tipo de archivos y resumir dicha información para extraer la que es realmente importante, en primera instancia, para el usuario. Esta técnica es especialmente útil cuando se producen errores en un sistema, ya que permite que el administrador o desarrollador del sistema pueda encontrar el origen del problema de una forma más fácil y rápida y, a partir de ahí, desencadenar una serie de respuestas para solucionarlo.

Además, el hecho de que la mayor parte de la información relevante de los ordenadores, así como los distintos eventos que se producen en estos y en los programas instalados en ellos, se encuentre presente en forma de *logs*, hace que el interés en el desarrollo de esta técnica sea aún mayor.

Por ello, son muchas las herramientas que han surgido con el fin de realizar este análisis de la forma más completa y eficiente posible. Así, se podría definir como un sistema completo de análisis de *logs* a aquel que ofrece las siguientes funcionalidades:

- Leer cada una de las líneas presentes en los *logs*, eliminar la información redundante y filtrar la información relevante para su conversión en campos o estructuras de datos.
- Guardar la información leída de los *logs* en algún tipo de almacén de datos (p.ej.: bases de datos, *clústers*, etc).
- Acceder al almacén de datos previo y ofrecer una visualización y/o monitorización en tiempo real de la información contenida de una forma diferente a la presente en los *logs* (como gráficos o diagramas).

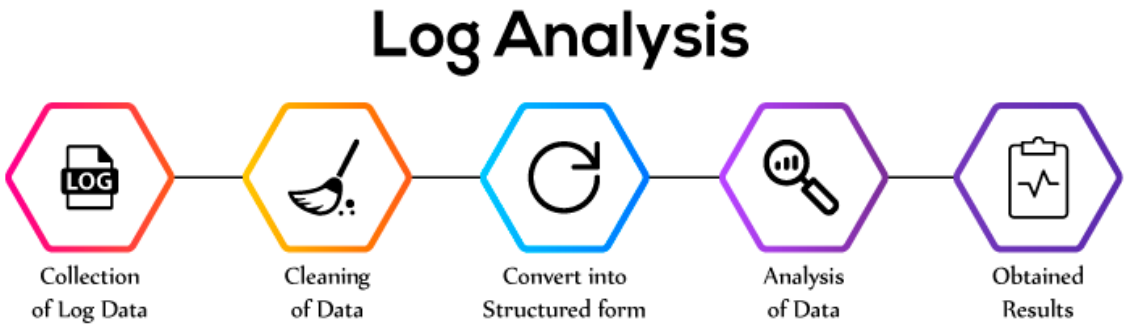


Ilustración 1. Proceso del análisis de datos

De esta forma, entre los productos existentes para el análisis de *logs*, se encuentran aquellos que se basan en una o dos de las funcionalidades anteriores (pero no las tres) y que, por tanto, suelen utilizarse junto a otros productos o programas para poder ofrecer un análisis más completo a los usuarios, y aquellos que incluyen las tres funcionalidades integradas en el mismo producto, no siendo necesario su despliegue junto a otros programas para una extensión de sus funcionalidades.

Dentro de ese primer grupo se encuentra la solución propuesta en este trabajo: los componentes del *ELK Stack* (Elasticsearch, Logstash y Kibana) que, aunque son tres productos diferentes, se encuentran bajo el nombre de una única compañía, Elastic. Por otro lado, dentro de los programas que incluyen por sí mismos las funcionalidades mencionadas previamente, se encuentran Sumo Logic o Splunk.

A lo largo de los siguientes apartados, se explicarán las características de estas dos herramientas, Sumo Logic y Splunk, y se completará una tabla que compare estas soluciones alternativas con la solución propuesta, explicando las razones que llevaron a la elección de dicha propuesta para la realización de este trabajo.

2.2. SUMO LOGIC

Ilustración 2. Logo de Sumo Logic

Sumo Logic es un servicio de análisis de datos de máquina para la administración de *logs* en tiempo real basado en el almacenamiento en la nube. Su interfaz, junto a la multitud de configuraciones que ofrece, hace que la puesta en marcha del sistema de agregaciones de *logs* sea una tarea mucho más fácil y rápida.

Entre las principales características de Sumo Logic se encuentran:

- Incluye una plataforma que unifica todos los *logs* y parámetros, analiza todos los datos y monitoriza las aplicaciones y la infraestructura en tiempo real.
- Éxito probado de los clientes, que confían en Sumo Logic para cubrir sus ideas de negocio.
- Arquitectura de múltiples usuarios que escala según la demanda para ayudar al rápido crecimiento del servicio y a la migración de datos a la nube.
- Constituye un punto de referencia en el establecimiento de seguridad Saas (*Software-as-a-service*).

Para acceder a la información presentada en este apartado y a información adicional acerca de Sumo Logic, se puede utilizar la referencia [1] de la bibliografía.

2.3. SPLUNK

Ilustración 3. Logo de Splunk

Splunk surge como un *software* capaz de buscar, analizar y monitorizar aquellos datos que tienen origen en las máquinas (*Big Data*), con el objetivo de abordar los mayores retos en la seguridad y las infraestructuras IT (*Information Tecnology*) más potentes. Actualmente se conoce como una de las mayores y más completas herramientas en el campo del análisis y manejo de *logs*.

Entre las principales características de Splunk se encuentran:

- Respuestas en tiempo real a través de gráficas y diagramas, con el objetivo de satisfacer las expectativas de los clientes y las metas de los negocios.
- Arquitectura que escala horizontalmente, a lo largo de una serie de servidores para servir como soporte a un gran número de usuarios y volúmenes de datos, y verticalmente, aumentando la velocidad de búsqueda e indexación y la capacidad de aprovechamiento de la CPU.
- Numerosos tutoriales y una rica documentación para los nuevos usuarios y que permiten una fácil y rápida puesta en marcha y configuración.

Para acceder a la información presentada en este apartado y a información adicional acerca de Splunk, se puede utilizar la referencia [2] de la bibliografía.

2.4. COMPARACIÓN DE LAS DISTINTAS SOLUCIONES Y LA PROPUESTA

Por último, y para acabar con el actual capítulo, en este apartado se va a presentar una tabla que permita una comparación de diferentes aspectos de los productos presentados (Sumo Logic y Splunk) y de la propuesta elegida.

Además, en esta tabla se compararán las versiones de uso gratuito de cada uno de los productos de cara a conocer la mejor opción sin utilizar las versiones más caras. Estas versiones serán: Sumo Logic *FREE*, Splunk *FREE* y *ELK Stack* (*Basic license*).

	Sumo Logic FREE	Splunk FREE	ELK Stack (Basic)
Recolección y envío de datos	✓	✓	✓
Almacenamiento de datos	✓	✓	✓
Búsqueda	✓	✓	✓
Análisis	✓	✓	✓
Visualización	✓	✓	✓
Servicio en la nube	✓	✗	✗
Limitaciones	500 MB/día	500 MB/día	Ilimitado
Plugins	✓	✓	✓
Integraciones	✓	✓	✓
Seguridad	✓	✗	✓
Control de acceso	✓	✗	✓
Documentación y tutoriales	✓	✓	✓
Comunidad y foros	Sólo lectura	Sólo lectura	Lectura y creación (número limitado)
Soporte	✗	✗	✗

Tabla 1. Comparación entre Sumo Logic, Splunk y ELK Stack

Con el objetivo de completar la información de la tabla anterior, se van a realizar algunas aclaraciones en cada uno de los aspectos comparados:

- **Capacidades de análisis de *logs***: aunque las tres soluciones incluyen diferentes métodos o herramientas para la recolección, envío, almacenamiento, búsqueda, análisis y visualización de los datos, Splunk es el que ofrece una mayor cantidad de opciones en estas áreas. Sin embargo, las numerosas limitaciones que reciben los tres productos en sus versiones gratuitas hacen que estas diferencias sean mucho menos apreciables.

- **Servicio en la nube**: Sumo Logic es un sistema basado principalmente en almacenamiento en la nube y, por lo tanto, incluye este servicio incluso en su versión gratuita. Por otro lado, tanto Splunk como el *ELK Stack* tienen versiones que permitirían este servicio (Splunk Cloud y la licencia *Enterprise*, respectivamente), pero esto incluiría un gran aumento en gastos, ya que se trata de sus licencias más caras.

- **Limitaciones**: mientras que Sumo Logic y Splunk tienen la limitación de poder almacenar como máximo 500 MB al día, dicha limitación no existe en el *ELK Stack*. Sin embargo, para conseguir esta ventaja, es necesario instalar la versión *Opensource* de *ELK Stack*, que obliga al usuario a establecer y configurar todos sus elementos por cuenta propia.

- ***Plugins* e integraciones**: las tres soluciones constituyen sistemas fácilmente integrables junto a otras herramientas. Con respecto a los *plugins*, es cierto que Splunk cuenta con más de 600 *plugins* disponibles, pero la suma de los *plugins* para cada uno de los productos que conforman el *ELK Stack* hace que se alcance, e incluso supere, dicha cantidad.

- **Seguridad**: Splunk no ofrece ningún sistema de seguridad a no ser que el usuario posea su versión más completa (*Enterprise*), mientras que Sumo Logic si incluye opciones de seguridad en su versión gratuita. Por otro lado, aunque *ELK Stack* no incluye opciones de seguridad a no ser que se posea su licencia más cara (*Enterprise*), la agregación del plugin de Search Guard permite la habilitación de algunas opciones para la seguridad del sistema.

- **Control de acceso**: siguiendo la idea de la anterior característica, Splunk no ofrece un sistema de control de acceso en su versión gratuita mientras que Sumo Logic si la ofrece. De la misma forma, la agregación de Search Guard en el *ELK* Stack permite la habilitación de un control de acceso tanto en el acceso a Elasticsearch como a la interfaz web de Kibana.

- **Documentación y tutoriales**: aunque las páginas web de cada uno de los productos ofrecen tutoriales y documentación para la puesta en marcha de cada uno de los sistemas, se considera que la documentación de Sumo Logic es mucho más pobre en comparación con la de Splunk, que se lleva la medalla de oro en este aspecto.

- **Comunidad y foros**: los usuarios tienen acceso a los temas publicados en los foros en cualquiera de las opciones. Sin embargo, es *ELK Stack* la única opción que permite la creación de entradas en el foro (para problemas, dudas, etc) por parte de usuarios con una licencia que no sea de pago.
- **Soporte**: las opciones de soporte por parte del equipo técnico están deshabilitadas en los tres productos en sus versiones gratuitas.

Como bien se ha dicho previamente, todas estas características se ven mejoradas en gran medida si se posee una licencia más cara de algunos de los productos, siendo Splunk el que más beneficios otorgaría al usuario en tal caso. Sin embargo, el coste de la compra de la licencia y el mantenimiento de los productos con dichas licencias los haría prácticamente insostenibles en una empresa que no tuviera una gran capacidad económica.

En el caso de este trabajo, el objetivo es buscar una solución que permita realizar de la forma más completa posible el análisis y administración de los logs, pero que lo haga de forma gratuita, para que el coste no sea una razón que impida su incorporación a la comunidad universitaria.

Es por esto, por la posesión de casi todas las características de la tabla de este apartado y por las escasas limitaciones en su versión gratuita, por la que se ha decidido tomar como solución a la proporcionada por ***ELK Stack***. Además, aunque uno de los problemas de esta solución es el incremento de esfuerzo necesario para su puesta en marcha y configuración, la solución propuesta en este trabajo incluye un sistema donde dichos procesos ya han sido realizados y acompañado de manuales que facilitan estos procesos, por lo que dicho esfuerzo dejaría de ser un inconveniente para esta solución.

3. INTRODUCCIÓN AL DISEÑO, ANÁLISIS, IMPLEMENTACIÓN E IMPLANTACIÓN

3.1. METODOLOGÍA USADA

El proyecto actual se ha basado en un resumen y adaptación de la metodología MÉTRICA v.3, de forma que, aunque se ha utilizado como referencia, muchos de los apartados presentes en dicha metodología no se han incluido en la actual memoria.

3.2. CICLO DE VIDA

El modelo de ciclo de vida elegido para el proyecto ha sido el modelo en cascada, en el que no se avanza a la siguiente etapa hasta que la actual no ha sido completada (en vez de trabajar en paralelo).

Además, se trata de un modelo con retroalimentación, de forma que se puede regresar a las etapas anteriores si se viera necesaria alguna modificación, eliminación o adicción al proyecto debido a la aparición de algún error o fallo. El diagrama siguiente representa el modelo de ciclo de vida elegido:

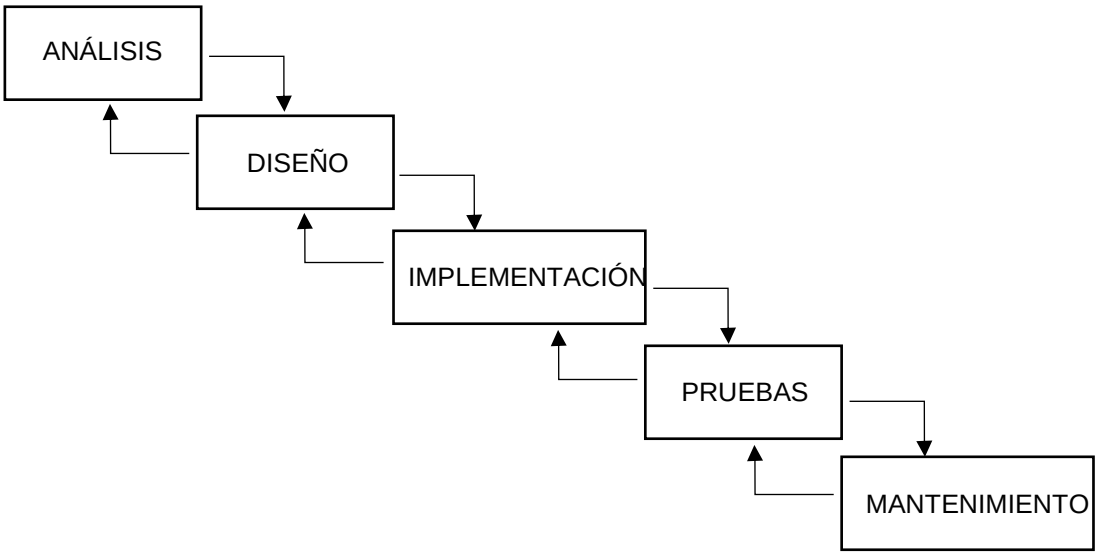


Ilustración 4. Ciclo de vida del proyecto

3.3. MARCO REGULADOR

En lo referido al marco regulador del proyecto, se han revisado las licencias de cada uno de los programas u herramientas incorporados en el sistema desarrollado (principalmente de Elastic) y se ha comprobado que no se incumple lo definido en dichas licencias.

Con respecto a la Ley de Protección de Datos, aunque es cierto que el sistema puede utilizar información de *logs* donde puede aparecer información relacionada con el usuario, como el caso de nombres de usuario en el archivo "*auth.log*", en ningún momento se accederá a su información privada ni al contenido de sus contraseñas.

Además, cualquier información recolectada por el sistema es únicamente visible para un usuario administrador, de forma que dicha información no se hace pública ni puede ser visualizada por un usuario corriente. Es por esto por lo que se considera que el proyecto actual cumple la Ley de Protección de Datos.

Por último, merece indicarse que el trabajo actual está sujeto a la licencia *Creative Commons* Reconocimiento – No Comercial – Sin Obra Derivada, que permite la descarga y compartición del trabajo, siempre y cuando se reconozca su autoría, pero no permite su comercialización ni la modificación de su contenido.

4. ANÁLISIS

En este capítulo se indicarán cuáles son los requisitos funcionales y no funcionales del sistema a desarrollar.

4.1. REQUISITOS

Para la definición de los requisitos del sistema, tanto funcionales como no funcionales, se utilizará la siguiente tabla como base:

RF-XX / RNF-XX			
Nombre	Nombre del requisito		
Prioridad	☐ Alta	☐ Media	☐ Baja
Necesidad	☐ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☐ Alta	☐ Media	☐ Baja
Verificabilidad	☐ Alta	☐ Media	☐ Baja
Descripción	Descripción del requisito		

Tabla 2. Plantilla de los requisitos

Con el objetivo de definir de la forma más completa y correcta los requisitos del sistema, cada una de estas tablas contiene información acerca de los siguientes atributos del atributo que representan:

- **Identificador:** código unívoco para cada requisito. Este está formado por:
  - **RF/RNF**: subcódigo que identifica si el requisito es funcional (RF) o no funcional (RNF)
  - **XX**: cifras numéricas que indican el número de requisito.
- **Nombre:** título representativo del requisito.
- **Prioridad:** atributo que define la urgencia con la que debe ser implementado un requisito. Tiene tres posibles categorías: alta, media y baja.
- **Necesidad:** atributo que representa la relevancia del requisito para los clientes y usuarios. Son tres las posibles categorías en este atributo: esencial (alta), deseable (media) y opcional (baja).

- **Estabilidad:** atributo que define la probabilidad con la que el requisito puede llegar a ser modificado en el tiempo de vida del sistema. Al igual que la prioridad, la estabilidad tiene tres categorías: alta, media y baja.
- **Verificabilidad:** atributo que representa la facilidad con la que se puede comprobar que el requisito se encuentra incluido en el sistema desarrollado. Tiene tres posibles categorías: alta, media y baja.
- **Descripción:** explicación general del requisito.

4.1.1. Requisitos funcionales

Los requisitos funcionales son aquellos que representan una capacidad o función que debe incluir el sistema a desarrollar. Se suelen identificar con el "qué" debe hacer el sistema.

En el caso del actual proyecto, los programas o herramientas que forman el sistema se dividirán en dos tipos de equipos:

- **Equipo servidor**: aquel desde el que se recoge y monitoriza la información del resto de equipos (clientes).
- **Equipo cliente**: aquel cuya información es procesada, enviada, almacenada y monitorizada por el equipo servidor.

Por esta razón, los siguientes requisitos funcionales se clasificarán según tengan relación con el equipo servidor, con los equipos cliente o con ambos (en caso de haberlos).

4.1.1.1. Requisitos funcionales del equipo cliente

RF-01			
Nombre	Envío al equipo servidor		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema recopilará información de los *logs* de los equipos cliente y la enviará al equipo servidor.		

Tabla 3. RF-01

RF-02			
Nombre	Permisos sobre los *logs*		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema deberá disponer de los permisos necesarios para poder leer y enviar los *logs* a monitorizar.		

Tabla 4. RF-02

RF-03			
Nombre	Múltiples equipos cliente		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema permitirá que varios equipos cliente envíen información al equipo servidor de forma simultánea.		

Tabla 5. RF-03

RF-04			
Nombre	Detección de nueva información		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema detectará, en el equipo cliente, cuándo son introducidas nuevas líneas en los *logs* que está monitorizando.		

Tabla 6. RF-04

4.1.1.2. Requisitos funcionales del equipo servidor

RF-05			
Nombre	Recepción de la información		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema, en el equipo servidor, se encargará de recibir la información recibida de los equipos cliente.		

Tabla 7. RF-05

RF-06			
Nombre	Filtrado de la información		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema también se encargará de filtrar la información recibida en el equipo servidor.		

Tabla 8. RF-06

RF-07			
Nombre	Almacenamiento de la información		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema almacenará la información recibida y filtrada de los equipos cliente.		

Tabla 9. RF-07

RF-08			
Nombre	Consulta de la información		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema permitirá la consulta de la información almacenada en el equipo servidor.		

Tabla 10. RF-08

RF-09			
Nombre	Visualización de la información		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema en el equipo servidor permitirá, por último, la visualización de forma gráfica de la información almacenada.		

Tabla 11. RF-09

RF-10			
Nombre	Monitorización del sistema de monitorización		
Prioridad	☐ Alta	☒ Media	☐ Baja
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema incluirá alguna herramienta o programa que permita la monitorización del propio sistema de monitorización para conocer sus gastos (CPU, memoria, etc).		

Tabla 12. RF-10

4.1.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos requisitos que definen las propiedades o características del sistema. Se suelen identificar con el "cómo" debe hacer algo el sistema.

En este apartado, los requisitos no funcionales también se dividirán según estén relacionados con el equipo servidor, los equipos cliente o ambos (en caso de haberlos).

Así, los requisitos no funcionales del sistema desarrollado son los siguientes:

4.1.2.1. Requisitos no funcionales del equipo cliente

RNF-01			
Nombre	Sistema operativo (equipo cliente)		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	Los equipos cliente deberán contar con un sistema operativo Debian 9.1 para poder acceder a los *logs* deseados.		

Tabla 13. RNF-01

4.1.2.2. Requisitos no funcionales del equipo servidor

RNF-02			
Nombre	Sistema operativo (equipo servidor)		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	Al utilizar contenedores de software para desplegar las herramientas, que se basan en el sistema operativo de la imagen que utilizan como base, el equipo servidor permitirá la instalación, configuración y despliegue del sistema en cualquier sistema operativo.		

Tabla 14. RNF-02

RNF-03			
Nombre	Sistema de monitorización en tiempo real		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema desarrollado permitirá una monitorización en tiempo real en el equipo servidor.		

Tabla 15. RNF-03

RNF-04			
Nombre	Control de acceso		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema incorporará un control de acceso a las herramientas, para evitar los accesos inadecuados o malintencionados.		

Tabla 16. RNF-04

RNF-05			
Nombre	Seguridad en el sistema de almacenamiento		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	El sistema incorporará mecanismos de seguridad para proteger los datos almacenados en el equipo servidor.		

Tabla 17. RNF-05

4.1.2.3. Requisitos no funcionales comunes

RNF-06			
Nombre	Despliegue en contenedores de software o de aplicación		
Prioridad	☒ Alta	☐ Media	☐ Baja
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Estabilidad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Descripción	Cada uno de los programas que forman parte del sistema, tanto en los equipos cliente como en el equipo servidor, serán desplegados mediante contenedores de software o de aplicación.		

Tabla 18. RNF-06

5. DISEÑO

5.1. SELECCIÓN DE LAS HERRAMIENTAS

En este apartado se realizará la discusión y comparación de las herramientas a usar en el sistema a desarrollar, conociendo las herramientas que lo forman al final del apartado.

5.1.1. Instalación local VS Contenedores Docker

En una primera instancia se pensó, como es usual, la instalación local en el sistema operativo de cada una de las herramientas que forma el sistema desarrollado. Sin embargo, en los últimos años se ha desarrollado una nueva forma de virtualización, la virtualización del sistema operativo, que merece ser tenida en cuenta. Esta es la virtualización basada en **contenedores de aplicaciones**.

Pero ¿qué son los contenedores de aplicaciones? Según [3]: “Los contenedores de aplicaciones son entornos ligeros de tiempo de ejecución que proporcionan a las aplicaciones los archivos, las variables y las bibliotecas que necesitan para ejecutarse, maximizando de esta forma su portabilidad.”

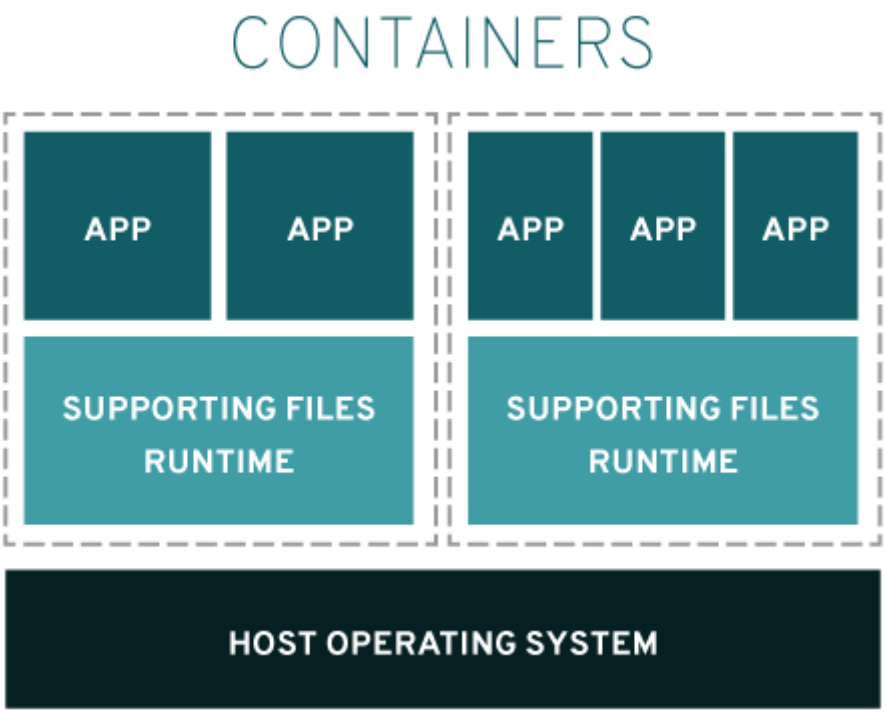


Ilustración 5. Virtualización basada en contenedores de software

Teniendo como referencia la información de [4], algunas de las principales características de los contenedores son:

- **Fácil instalación**: los contenedores se instalan a través de imágenes, que podrían definirse como unos pequeños sistemas operativos con algunas aplicaciones instaladas. Una vez el usuario asigna a la imagen la configuración deseada, el despliegue del contenedor puede resumirse en la ejecución de un único comando.
- **Gran portabilidad**: una vez que un contenedor es configurado, este puede ser fácilmente trasladado a cualquier otro sistema gracias a su gran independencia de la plataforma sobre la que es desplegado. El único requisito para ello es que el sistema destino debe soportar este tipo de tecnología.
- **Aplicaciones aisladas**: las aplicaciones instaladas en un determinado contenedor son completamente independientes de las instaladas en otros contenedores, por lo que, de esta forma, aplicaciones que tuvieran requerimientos diferentes, incluso opuestos, podrían funcionar correctamente en un mismo sistema.
- **Autosuficiencia**: siguiendo la definición previamente dada, una de las ventajas de los contenedores es que se crean únicamente con las librerías y archivos necesarios para que las aplicaciones que se ejecuten en ellos funcionen correctamente.

Por estas características anteriores, las cuáles son ventajas sobre la instalación local en el sistema operativo, se ha decidido que cada una de las herramientas del sistema desarrollado se instale y despliegue en contenedores diferentes. Además, se ha decidido el uso de **Docker** como herramienta para el control de estos contenedores, ya que facilita su creación y despliegue, entre otros.

5.1.2. Solr VS Elasticsearch

Entre los principales servidores de búsqueda (tanto almacenamiento como búsqueda de datos), destacan **Solr**, desarrollado por Apache, y **Elasticsearch**, perteneciente al ELK Stack. A continuación, se realizará una comparación de ambas herramientas, utilizando como referencia la información contenida en [5]:

	SOLR	ELASTICSEARCH
Formato de queries	XML, CSV, JSON	JSON
Formato de salida	JSON, XML, Python, PHP, Ruby, etc	JSON, HTML/XML
Librerías soportadas	Java	Java, PHP, Javascript, Python, .NET, etc
Lenguaje de análisis	Lucene	Lucene
Autonomía del clúster	No (depende de un servidor de Zookeper)	Si (nodos de Elasticsearch)
Cachés	Global (invalidado con cada cambio de segmento)	Por segmento (mejor para datos que cambien dinámicamente)
Velocidad de búsqueda	Mayor en datos estáticos	Mayor en datos dinámicos
Visualización de datos	Banana, Zepelin	Grafana, Kibana
Instalación y configuración	A través de una detallada documentación	Más intuitiva
Documentación	Muy bien documentado	Falta de documentación

Tabla 19. Comparación entre Solr y Elasticsearch

Observando la tabla anterior se puede ver en qué aspectos cada herramienta es mejor. Por un lado, Solr se caracteriza por el soporte de una gran cantidad de formatos de entrada y salida, una gran documentación y una mayor velocidad de búsqueda en datos estáticos. Por otro, Elasticsearch ofrece un mayor soporte de librerías, un clúster autónomo, una mayor facilidad de aprendizaje e instalación y una mayor velocidad de búsqueda en datos dinámicos.

Aunque cada herramienta tiene sus ventajas, se ha elegido **Elasticsearch** por su facilidad de aprendizaje e instalación y por la autonomía de su clúster (lo que impide la necesidad de un servidor externo como en el caso de Solr), además de por su pertenencia al ELK Stack, que proporciona numerosas herramientas útiles para el análisis completo de logs.

5.1.3. Fluentd VS Logstash

En cuanto al campo de la administración de logs (recolección, envío y filtrado de datos), son dos herramientas las que destacan: **Fluentd** y **Logstash**. A continuación, basándose en [6], se presentará una tabla comparativa de características de ambas herramientas.

	FLUENTD	LOGSTASH
Plataformas compatibles	Windows y Linux	Windows y Linux
Lenguaje de programación	CRuby (C)	JRuby (Java)
Visualización y monitorización	A través de herramientas externas (Grafana, Kibana)	A través de plugins
Repositorio de plugins	Descentralizado	Centralizado
Plugins disponibles	500+ (sólo 10 oficiales)	200+
Relación con Elasticsearch	Plugin	Pertenece al ELK Stack
Transportador de datos alternativo (*Data shipper*)	Fluentd Forwarder	Elastic Beats

Tabla 20. Comparación entre Fluentd y Logstash

Como se puede observar, ambas herramientas cubren las funcionalidades de un administrador de datos de formas diferentes (distintos lenguajes de programación, diferentes formas de visualización, etc), se ejecutan en las plataformas más comunes (Windows y Linux) y recaen en otras herramientas para el transporte de datos en sistemas más grandes.

Es por esta similitud de funcionalidades, y por su íntima relación con Elasticsearch y con el ELK Stack, por lo que se ha elegido **Logstash** como herramienta para la administración de datos, pero esto no significa que Fluentd tuviera un gran inconveniente que impidiera su uso.

5.1.4. Grafana VS Kibana

Dentro de los visualizadores de datos, utilizados para la representación gráfica de los mismos una vez se encuentran almacenados en bases de datos o *clusters*, son dos herramientas las más importantes: **Grafana** y **Kibana**, perteneciente al ELK Stack. En la siguiente tabla se compararán ambas herramientas, usando como referencia la información de [7]:

	GRAFANA	KIBANA
Instalación y configuración	Fácil	Fácil
Propósito	Análisis de métricas (CPU, uso de disco, memoria, etc)	Análisis de logs
Integraciones	Prometheus, MySQL, Graphite, InfluxDB, etc	Elasticsearch
Control de acceso	Incluido por defecto	Con X-Pack o Search Guard
Gráficos y visualizaciones	Mayor rango de personalización	Mayor variedad de tipos de gráficos
Sistema de alertas	Incluido por defecto	Sólo con X-Pack

Tabla 21. Comparación entre Grafana y Kibana

En este caso la elección entre las dos herramientas parece ser más fácil que en los casos anteriores. Si bien Grafana tiene un mayor número de integraciones posibles e incluye por defecto sistemas de control de acceso y alerta, Kibana es la única opción que ofrece análisis de logs (en vez de métricas como el caso de Grafana).

Además, el hecho de que Kibana pueda integrarse únicamente con Elasticsearch puede verse más como una ventaja que como un inconveniente en este caso, ya que al pertenecer el resto de herramientas elegidas al ELK Stack, es una ventaja que el visualizador de datos también pertenezca al mismo grupo.

Por otro lado, aunque Kibana no incluye por defecto el control de acceso o algún sistema de autentificación, este puede ser obtenido a través de *plugins* como Search Guard o X-Pack, los cuales se compararan en un apartado posterior dentro de este mismo capítulo.

Estas son las principales razones por la que se ha elegido a **Kibana** como el visualizador de datos del sistema.

5.1.5. ¿Logstash y/o Filebeat?

Como bien se mencionó previamente, Logstash actuaría como un administrador de logs dentro del sistema desarrollado, encargándose de la recolección, envío y filtrado o transformación de la información contenida en dichos ficheros.

Sin embargo, uno de los principales problemas que tiene Logstash es que, al necesitar JVM (Java Virtual Machine) y estar implementado en el lenguaje de programación Ruby, su consumo de memoria es bastante elevado. Este problema se agrava exponencialmente cuanto mayor es el número de pipelines (tuberías para el envío de datos) usados, siendo necesaria un pipeline por cada sistema cuyos logs se quieran administrar (es decir, un pipeline por ordenador a monitorizar).

Teniendo en cuenta que el actual trabajo está destinado a una monitorización de los ordenadores de un aula informática, el número de pipelines que tendrían que estar activas sería bastante elevado y, por tanto, el gran consumo de memoria de Logstash se convertiría en un problema prácticamente insostenible. Es en este momento cuando surge la idea del uso de transportadores de datos (o *data shippers*).

El origen de los transportadores de datos tuvo lugar con Lumberjack y, posteriormente, surgió Logstash Forwarder, con el principal y único objetivo de enviar los datos a Logstash y así reducir el consumo de memoria del mismo. Sin embargo, años más tarde, Elastic, la empresa desarrolladora del ELK Stack, dio lugar a un conjunto de herramientas para cubrir esta misma función: los **Beats**.

Este conjunto de herramientas se basa en el envío de información a través de latidos o *beats* (origen del nombre del conjunto). Entre las herramientas incluidas en el conjunto se encuentran Metricbeat (envío de métricas), Packetbeat (envío de datos de red) o Winlogbeat (envío de logs de los eventos de Windows). Pero la herramienta que ha interesado más de cara a su incorporación en el sistema a desarrollar es **Filebeat**, herramienta destinada al envío de archivos y, por tanto, de logs.

Las dos principales ventajas que ofrece Filebeat sobre Logstash [8] son las siguientes:

- Se trata de una herramienta ligera, que no consume mucha memoria.
- Soporta cifrado TLS y SSL, además de que por sí mismo ya es extremadamente seguro.

Ahora bien, ¿por qué incluir Filebeat “junto a” y no “en vez de” Logstash? Como bien se ha explicado, Logstash necesita de un transportador de datos como Filebeat para reducir en gran medida su consumo de memoria, sobre todo en redes con un gran número de ordenadores. Pero, por otro lado, Filebeat también necesita a Logstash, ya que, aunque el primero ofrece la recolección y envío propios de un administrador de logs, el segundo ofrece un gran sistema de filtrado o transformación de los mismos, necesario para una correcta visualización en Kibana.

Por tanto, y para poder solucionar ese gran problema de Logstash, el cuarto elemento incluido en el sistema desarrollado, junto a los componentes del ELK Stack (Elasticsearch, Logstash y Kibana), es **Filebeat**.

5.1.6. ¿X-Pack y/o Search Guard?

Tras las discusiones y comparaciones de los anteriores apartados las herramientas que forman el núcleo funcional del sistema son Elasticsearch, Logstash, Kibana y Filebeat, aparte de Docker para el despliegue de cada una de las herramientas.

Al ser muchas herramientas y haber distintos accesos entre ellas, hay una característica importante que merece tenerse en cuenta en el sistema actual: la seguridad. Y es en este aspecto donde Elastic ofrece un componente en forma de plugin para ofrecer una serie de servicios adicionales al *ELK Stack*: **X-Pack**.

Entre los principales servicios de seguridad que ofrece X-Pack se encuentra la habilitación de un control de acceso basado en roles para los componentes del *ELK Stack* y el cifrado de las comunicaciones entre dichos componentes y entre los nodos del clúster de Elasticsearch usando SSL o TLS. Sin embargo, el acceso a estos servicios que ofrece X-Pack requiere una licencia de pago en Elastic, algo que no interesa incorporar al trabajo actual por tratarse una búsqueda de un sistema gratuito, libre de licencias de pago. Surge de esta forma una segunda opción para mejorar la seguridad del sistema: **Search Guard**.

Search Guard ofrece los mismos servicios comentados previamente (cifrado SSL/TLS y control de acceso basado en roles) pero, al contrario que X-Pack, estos se encuentran incluidos en la versión gratuita, por lo que no es necesaria la adquisición de una licencia de pago para poder disfrutar de dichos servicios. Además, Search Guard posee una documentación detallada que facilita su instalación y configuración.

Por lo tanto, la herramienta que se va a utilizar para mejorar la seguridad del sistema es Search Guard, principalmente por no requerir una licencia de pago para disfrutar de sus servicios.

Por otro lado, aunque X-Pack no proporciona ningún servicio de seguridad sin disponer de una licencia de pago, sí que ofrece un sistema de monitorización (uso de CPU, memoria, etc) de los componentes del *ELK Stack* con la licencia básica, que se obtiene de forma gratuita y con una duración de un año (renovable una vez acabado dicho período). Si bien el sistema desarrollado tiene como objetivo la monitorización de los logs de una serie de equipos informáticos, es interesante incluir esta herramienta para, por así decirlo, “monitorizar el sistema de monitorización”.

Entonces, y finalizando con ello la elección de herramientas, las últimas dos herramientas incorporadas al sistema son **X-Pack**, para la monitorización del *ELK Stack*, y **Search Guard**, para mejorar la seguridad.

5.2. DISEÑO DE LA PROPUESTA

Mientras que en el apartado anterior se mostraron las características comparables entre cada par de herramientas, en este apartado se presentará cada una de las herramientas elegidas y se mencionarán o explicarán algunas de sus principales funcionalidades. Además, al final del apartado se presentará la arquitectura final del sistema.

5.2.1. Docker

Ilustración 6. Logo de Docker

Docker [9] es una plataforma de software que gestiona y facilita la creación, eliminación y despliegue de contenedores de aplicaciones. Permite reducir el despliegue de un contenedor a la ejecución de un único comando con una serie de parámetros o añadiendo dichos parámetros a un archivo común de la plataforma llamado "*Dockerfile"*.

Además, Docker ofrece una serie de aplicaciones para ampliar sus funcionalidades. Las principales son:

- **Docker Machine**: permite la creación de "máquinas" (o *machines)* para la simulación de distintos entornos de operación desde un mismo ordenador.
- **Docker Compose**: permite el despliegue simultáneo de varias aplicaciones, cada una en su correspondiente contenedor aislado. Igual que para desplegar un solo contenedor se utilizan archivos Dockerfile, para el despliegue conjunto mediante esta aplicación se utilizan archivos llamados "*docker-compose.yml*".
- **Docker Swarm**: permite el uso de un "modo *Swarm*", presente en Docker, para facilitar el control de varios hosts que cuenten con Docker.

Por otro lado, algunos de los principales comandos que ofrece Docker son:

- `docker run`: para desplegar un contenedor a partir de una imagen.
- `docker ps`: para mostrar todos los contenedores en ejecución.
- `docker-compose up/down`: para desplegar los contenedores del "*docker-compose.yml*" existente en el directorio actual.
- `docker-compose build`: para (re)compilar los contenedores del docker-compose.yml existente en el directorio actual.

5.2.2. Elasticsearch

Ilustración 7. Logo de Elasticsearch

Elasticsearch [10] es un servidor de búsqueda de código abierto que pertenece a la compañía Elastic. Entre sus principales características se encuentran que fue creado sobre Apache Lucene, está escrito en Java, tiene una API RESTful y está basado en JSON.

No sólo forma parte del ELK Stack (se corresponde con la 'E' de dicho nombre), sino que se considera el punto central de dicho conjunto, ya que tanto Logstash como Kibana requieren el correcto funcionamiento de esta herramienta para almacenar datos de logs o visualizarlos.

Elasticsearch está basado en un clúster, el cual se divide físicamente en nodos y lógicamente en índices, siento estos últimos a los que accede Kibana para acceder a los datos almacenados en el clúster. Además, permite el uso de comandos, tanto en la consola de comandos como en la consola de Kibana, para, por ejemplo, comprobar el estado del clúster o administrar los índices existentes.

5.2.3. Logstash

Ilustración 8. Logo de Logstash

Logstash [11] es una herramienta de código abierto creada para la administración de logs basada en un sistema *input-filter-output*, es decir, recoge los datos de los logs o los recibe de una herramienta externa, les aplica un filtrado (opcional) y se los reenvía a otra herramienta que se encargue de su almacenamiento o visualización. Al igual que Elasticsearch, pertenece al ELK Stack, correspondiéndose con la 'L' de dicho conjunto.

Aunque en el sistema actual es Filebeat el que se encarga de la recolección y envío de los logs (unas de las principales funcionalidades de Logstash), Logstash incluye una serie de potentes *plugins* para el filtrado de estos archivos, lo que le hace seguir teniendo un importante rol en el análisis de logs. Uno de los filtros más importante que incluye Logstash es el **filtro *grok***.

5.2.3.1. Filtro grok

El filtro *grok* se encarga de recibir una línea de un *log*, entre otros tipos de entradas, y, si dicha línea coincide con un determinado formato, almacenar la información marcada como relevante en una serie de campos para facilitar su búsqueda y visualización una vez se encuentre almacenado. De hecho, sin el uso de este filtro, la línea entera del *log* se almacenaría en un único campo llamado *message*.

La explicación del filtro podría ser mucho más clara utilizando un ejemplo: en la siguiente imagen se puede observar, marcada en gris, una línea procedente de un log donde el primer elemento es una dirección IP. De forma paralela, se puede observar en la configuración del filtro que esta comienza con un "%{IP:client}, donde:

- "%{ }" indica la aparición de un dato a almacenar en un campo.
- "IP" indica que se va a leer un dato estructurado como una IP.
- "client" indica el campo donde almacenar el dato leído.

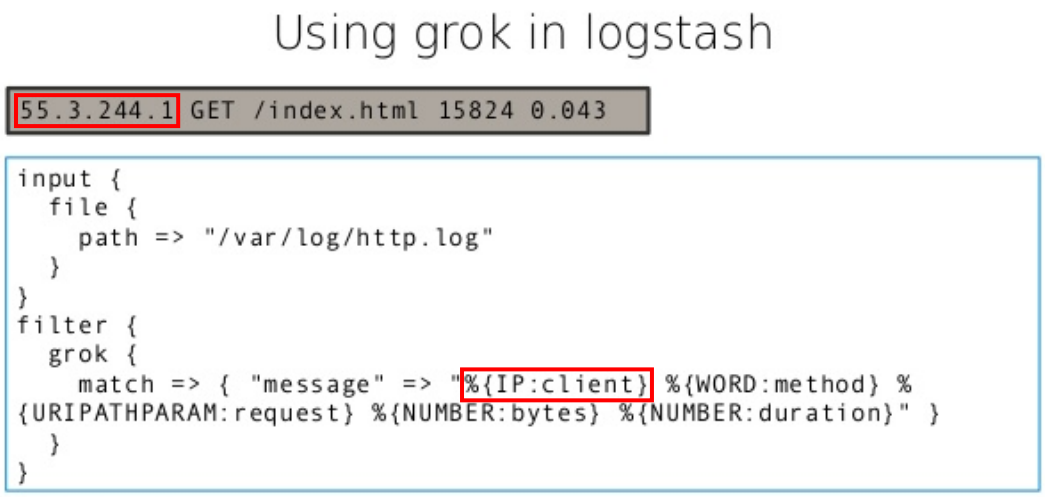


Ilustración 9. Ejemplo del filtro grok en Logstash

De esta forma, si se recibe una línea de un *log* que tenga la estructura especificada en el filtro, como es el caso del ejemplo marcado en gris, se obtendrá el campo completado "client: 55.3.244.1", entre otros campos (*method*, *request*, *bytes* y *duration*).

5.2.4. Kibana

Ilustración 10. Logo de Kibana

Kibana [12] es un visualizador de datos de código abierto que permite la representación de la información contenida en los índices de Elasticsearch utilizando visualizaciones (gráficos) y *dashboards* (conjuntos de gráficos), los cuáles pueden ser creados y configurados a través de la interfaz gráfica que proporciona. Kibana es el tercer y último elemento del *ELK Stack*, coincidiendo con la 'K' de dicho nombre.

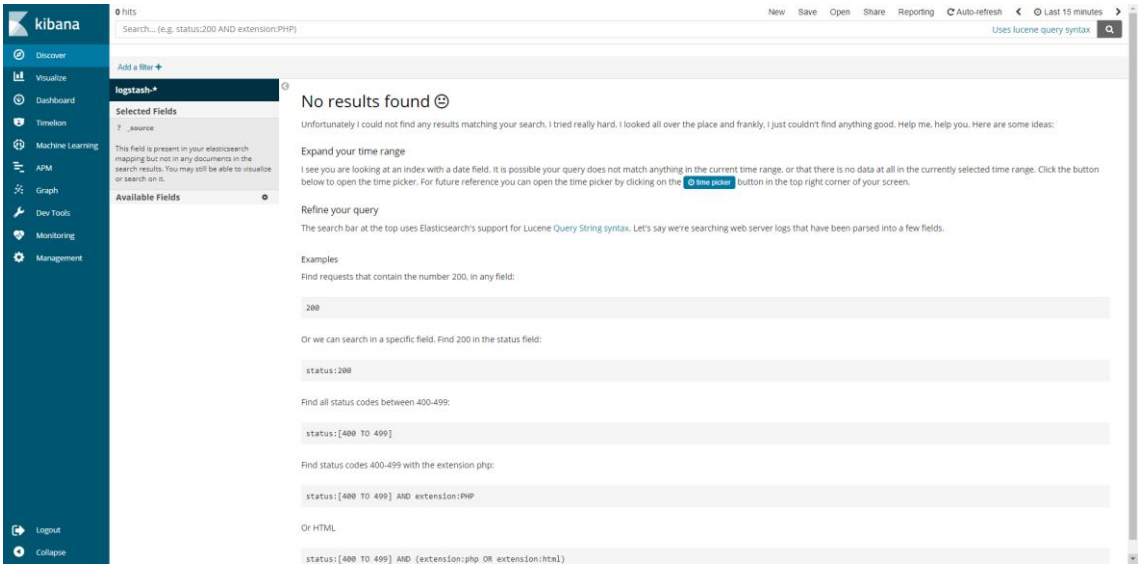


Ilustración 11. Interfaz gráfica de Kibana

Además de una interfaz gráfica, Kibana ofrece una serie de complementos o funcionalidades para la visualización de datos y la interacción con Elasticsearch, los cuáles se presentarán brevemente en los siguientes subapartados.

5.2.4.1. Parámetros para la visualización

Kibana incluye dos parámetros para controlar la visualización de los datos:

- **Actualización automática**: permite especificar un periodo de tiempo para que Kibana se actualice y compruebe si hay nueva información en los índices. Se encuentra desactivado por defecto.

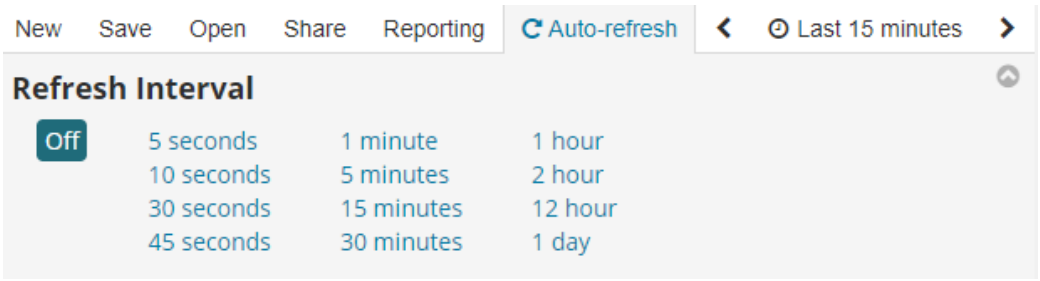


Ilustración 12. Actualización automática de Kibana

- **Rango de tiempo**: permite especificar un rango de tiempo, relativo o absoluto, para mostrar únicamente la información de Elasticsearch que pertenezca a dicho rango.

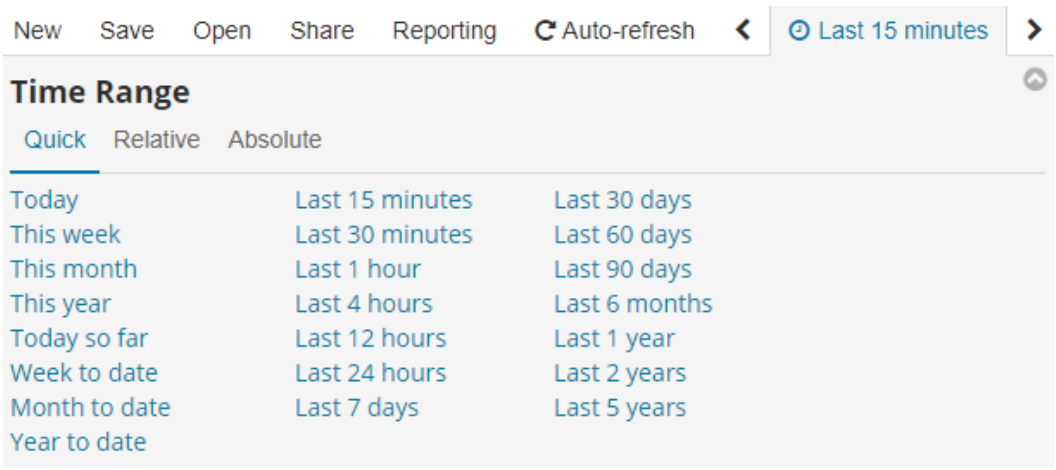


Ilustración 13. Rango de tiempo en Kibana

5.2.4.2. Consola de comandos

Kibana proporciona una consola de comandos propia para interactuar con Elasticsearch mediante la inserción de *queries* (consultas). A este complemento se accede a través de la opción "*Dev Tools*" de la barra de navegación de Kibana.

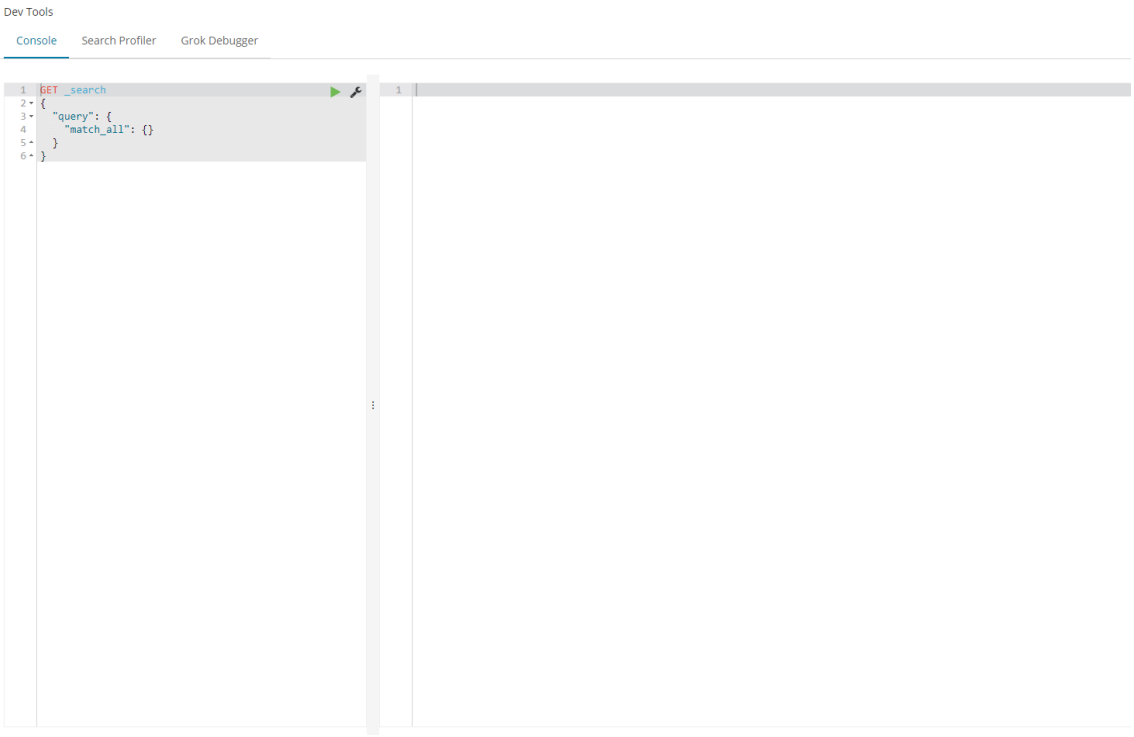


Ilustración 14. Consola de comandos de Kibana

5.2.4.3. Tipos de visualizaciones

Por último, merecen indicarse los múltiples tipos de visualizaciones que ofrece Kibana para la representación de los datos, las cuáles pueden ser accedidas a travésde la opción "*Visualize*" de la barra de navegación.

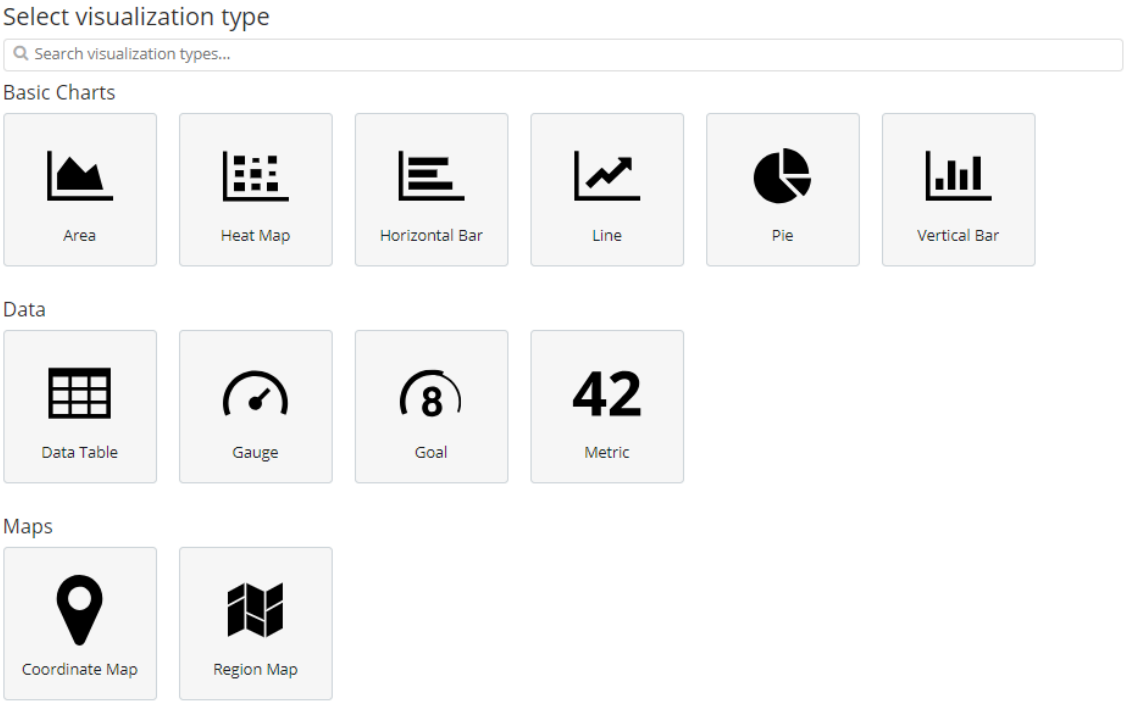


Ilustración 15. Tipos de visualizaciones de Kibana (I)

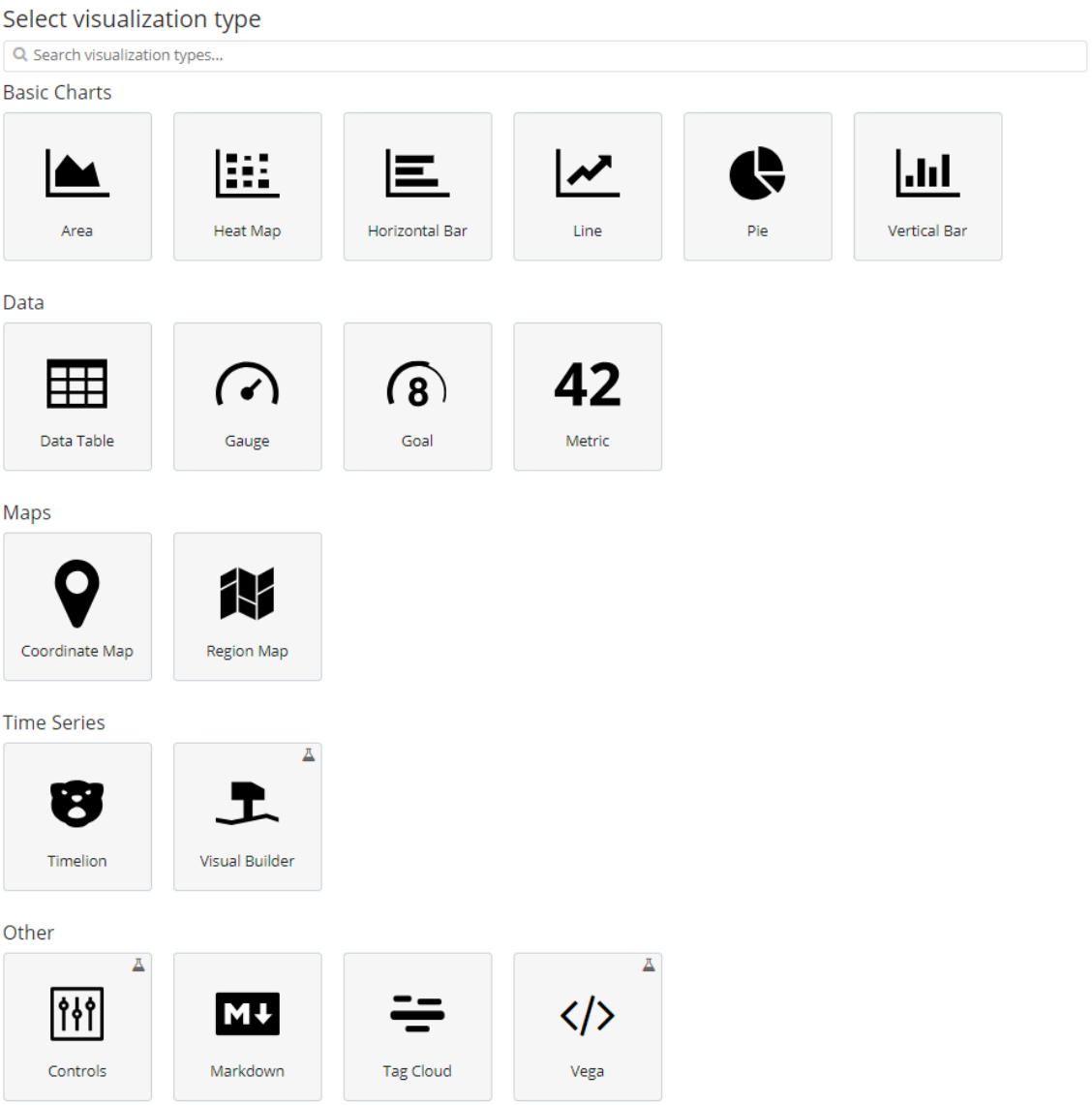


Ilustración 16. Tipos de visualizaciones de Kibana (II)

5.2.5. Filebeat

Ilustración 17. Logo de Beats

Filebeat [13] es un transportador de datos (*data shipper*) que pertenece al grupo Beats de Elastic, donde se encarga del envío de información relacionada con ficheros (principalmente *logs*) a través de pulsos o *beats*. Al formar parte de los productos de Elastic, Filebeat se convierte en una herramienta fácilmente configurable para trabajar junto a Logstash, Elasticsearch y Kibana (*ELK Stack*).

Filebeat se basa en el uso de dos tipos de elementos: ***harvesters*** (o "cosechadoress"), encargados de leer el contenido de un único archivo, y ***prospectors*** (o "buscadores"), encargados de administrar los *harvesters* y encontrar los archivos que tienen que leer estos.

Además, los *harvesters* tienen un parámetro para especificar cada cuanto tiempo deben comprobar si se han añadido nuevas líneas en los archivos que están leyendo, siendo este tiempo 10 segundos por defecto.

5.2.6. X-Pack

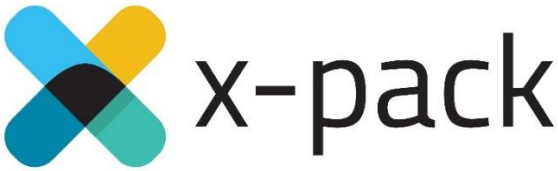


Ilustración 18. Logo de X-Pack

X-Pack [14] es una extensión del *ELK Stack* que lo complementa añadiendo una serie de servicios como monitorización, seguridad o un sistema de alertas e informes, lo que le hace el mejor plugin para completar el *stack*.

El principal servicio que ofrece X-Pack es el de la seguridad, permitiendo un cifrado de las comunicaciones entre los elementos del *stack* usando SSL/TLS y utilizando un control de acceso basado en roles en cada una de las herramientas. Pero, tal y como se mencionó en el apartado de comparación de herramientas de este mismo capítulo, el principal problema de X-Pack es la necesidad de una licencia de pago en Elastic (*Gold*, *Platinum* o *Enterprise*) para poder disfrutar de la mayoría de los servicios, a excepción de la monitorización, la cual es visible en la interfaz de Kibana.

Además, al incluir X-Pack en el sistema, se obtienen un par de complementos en Kibana, además de la monitorización.

5.2.6.1. Monitorización

Al activar X-Pack, se desbloqueará la opción "*Monitoring*" en la barra de navegación de Kibana. Desde esta opción, se puede observar el consumo de memoria o CPU, entre otros, de Logstash, Elasticsearch y Kibana, así como el estado de cada uno.

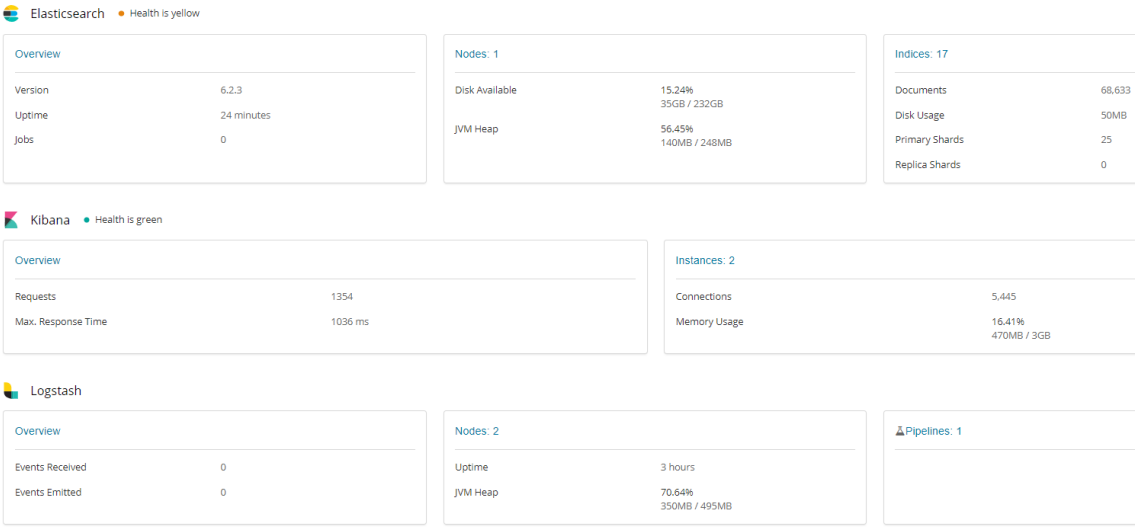


Ilustración 19. Monitorización de X-Pack en Kibana

5.2.6.2. Search Profiler

A través del *Search Profiler,* se puede conocer el tiempo que tarda Kibana en realizar una determinada consulta a todos o determinados índices de Elasticsearch. A este complemento se accede a través de la opción "*Dev Tools*" de la barra de navegación de Kibana, una vez X-Pack se encuentra instalado en el sistema.

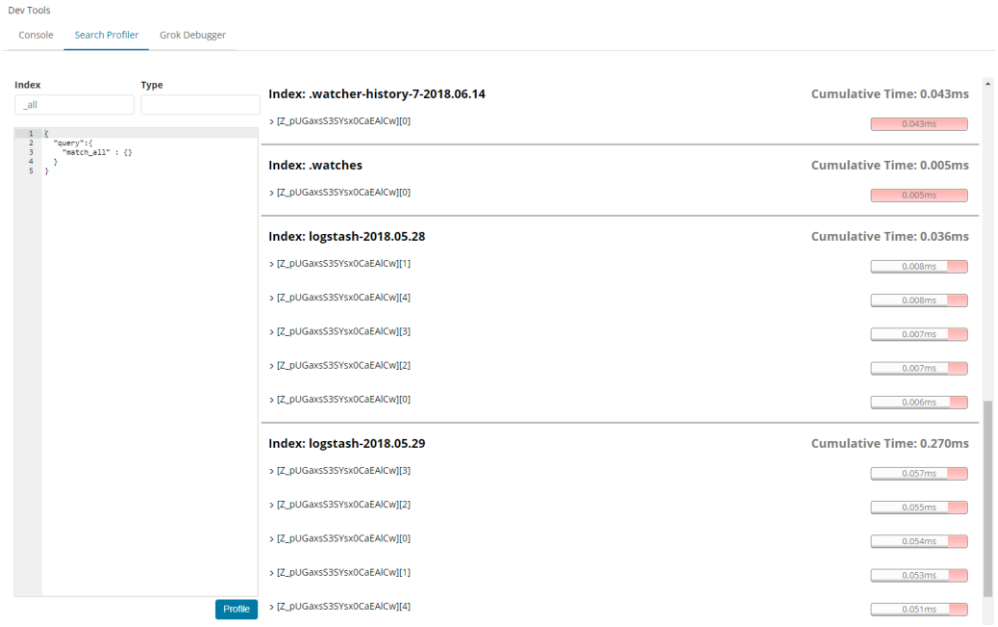


Ilustración 20. Search Profiler en Kibana

5.2.6.3. Grok Debugger

X-Pack desbloquea en Kibana el *Grok Debugger* para facilitar el uso del filtro *grok* de Logstash, conociendo el resultado que se obtendrá tras aplicar una determinada configuración del filtro sobre una o varias líneas de texto. A este complemento se accede a través de la opción "*Dev Tools*" de la barra de navegación de Kibana, una vez X-Pack se encuentra instalado en el sistema.

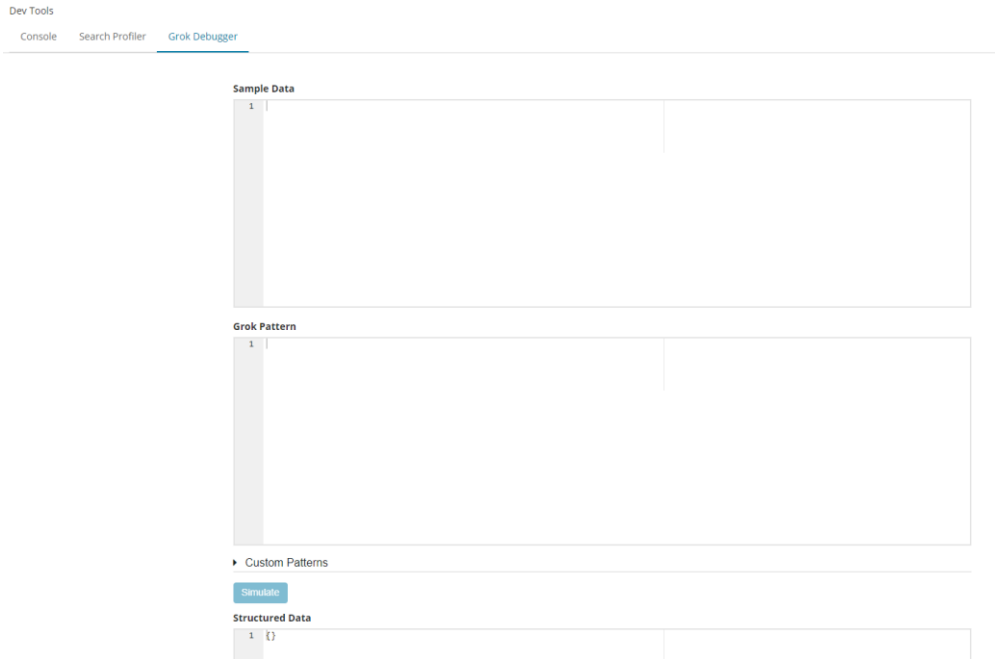


Ilustración 21. Grok debugger

5.2.6.4. Control de acceso basado en roles

X-Pack ofrece un control de acceso basado en roles, el cual puede ser configurado (tanto usuarios como roles) en la interfaz de Kibana, a través de la opción "*Management"* situada en la barra de navegación de Kibana.

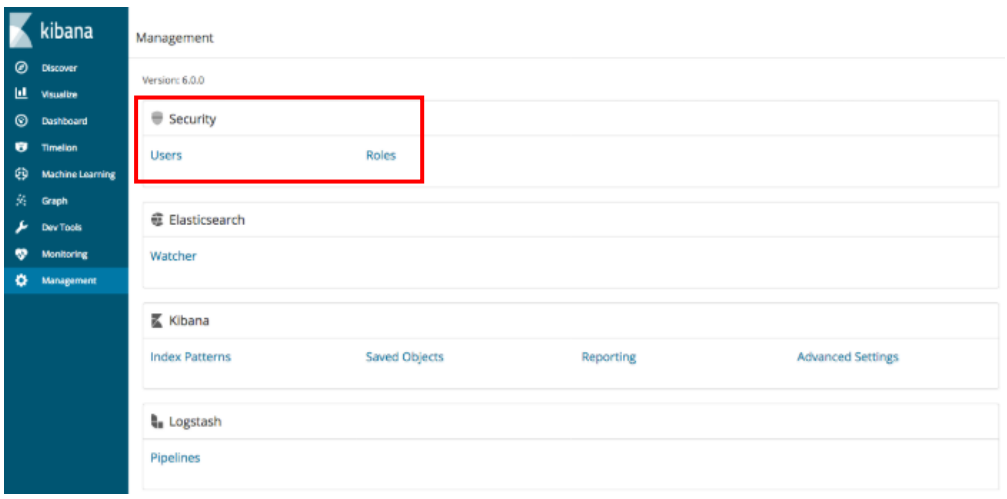


Ilustración 22. Control de acceso basado en roles

Como ya se ha mencionado en este apartado, este servicio solamente se desbloquea en Kibana si se instala X-Pack con una licencia de pago en Elastic (*Gold*, *Platinum* o *Enterprise*).

5.2.7. Search Guard

Ilustración 23. Logo de Search Guard

Search Guard [REFERENCIA] es un plugin que, aunque no pertenece a la compañía de Elastic, fue desarrollado para suministrar un servicio de seguridad a Elasticsearch y al resto de componentes. Así, Search Guard provee un encriptado TLS/SSL entre los nodos que forman el clúster y, de forma opcional, entre las herramientas del *ELK Stack* (Elasticsearch, Logstash y Kibana) permitiendo el acceso via HTTPS.

Además, Search Guard incluye un control de acceso basado en roles con la misma filosofía que el de X-Pack, pero con la diferencia de que este no puede configurarse a través de la interfaz de Kibana, sino usando un fichero de configuración. El uso de Search Guard implica por tanto la modificación de los archivos de configuración de las herramientas del ELK Stack para su intercomunicación y la introducción de credenciales cada vez que se accede a Kibana.

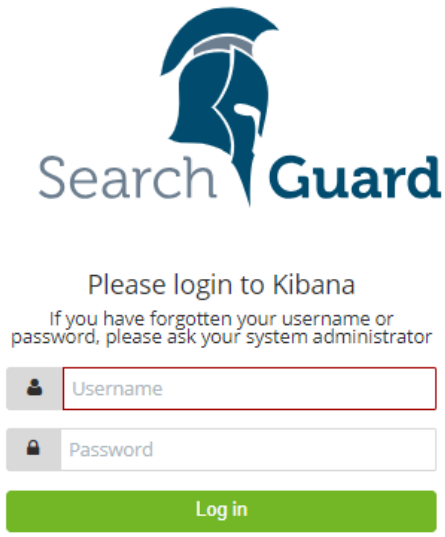


Ilustración 24. Inicio de sesión para acceder a Kibana

5.2.8. Arquitectura del sistema desarrollado

Tras presentar cada una de las herramientas usadas en el sistema, en este apartado se presentará la arquitectura general del sistema.

Para la realización de la monitorización deseada es necesaria la distinción entre dos tipos de ordenadores, según su función en el proceso de análisis de logs:

- **Equipo(s) cliente**: uno o varios equipos cuyos logs son procesados para ser monitorizados en el equipo servidor.
- **Equipo servidor**: aquel donde se realiza la monitorización de los equipos cliente.

De esta forma, cada tipo de ordenador contará con un conjunto de herramientas distintas. Como caso excepcional, todo equipo informático que sea incluido en el sistema, sea cliente o servidor, deberá tener instalado Docker, con el objetivo de poder desplegar cada herramienta en un contenedor de software distinto.

En el lado del equipo cliente únicamente se instalará Filebeat, ya que en estos equipos la única labor que se realizará será el acceso a los logs y el envío de la información contenida en estos al equipo servidor.

En el caso del equipo servidor, este incluirá el resto de las herramientas presentadas en este capítulo. Esto es: Logstash, Elasticsearch y Kibana para el funcionamiento básico del análisis de logs, y X-Pack y Search Guard para añadir los servicios de seguridad y monitorización al equipo.

De esta forma el resumen de las herramientas en cada tipo de ordenador sería el siguiente.

Equipo(s) cliente	Equipo servidor
• Docker • Filebeat	• Docker • Logstash • Elasticsearch • Kibana • X-Pack • Search Guard

Tabla 22. Herramientas en cada tipo de equipo

Con respecto a los *logs* que se monitorizan con el sistema desarrollado, el principal *log* que se monitorizará es *"auth.log*", situado por defecto en el directorio "*/var/log"* y que contiene información acerca de los inicios de sesión o los errores producidos durante los mismos: fallo de contraseña, usuario inexistente, etc.

Por otro lado, con respecto al proceso de análisis de los logs, es decir, el proceso que se sigue para el envío de información de extremo a extremo (desde Filebeat en el equipo cliente hasta Kibana en el equipo servidor), un resumen del proceso sería el siguiente:

1) **Filebeat** genera un *harvester* para leer los logs indicados en su archivo de configuración y enviar cada línea de los mismos a Logstash.
2) **Logstash** recibe cada una de las líneas de los logs enviadas por Filebeat, aplica los filtros que tenga especificados en su archivo de configuración (si es que hay alguno) y envía el resultado a Elasticsearch.
3) **Elasticsearch** almacena los datos enviados por Logstash en los nodos (almacenamiento físico) del clúster y crea índices (almcenamiento lógico) para que Kibana pueda acceder a los datos almacenados.
4) **Kibana** accede a los índices de Elasticsearch para obtener los datos almacenados en dicho clúster y mostrarlos en su interfaz gráfica.
5) Una vez Kibana tiene una réplica de los datos, el **usuario** puede representarlos utilizando para ellos las visualizaciones y *dashboards* que proporciona Kibana.

Para aclarar en mayor medida la estructura del sistema y el proceso de análisis de los logs se puede utilizar el siguiente diagrama.

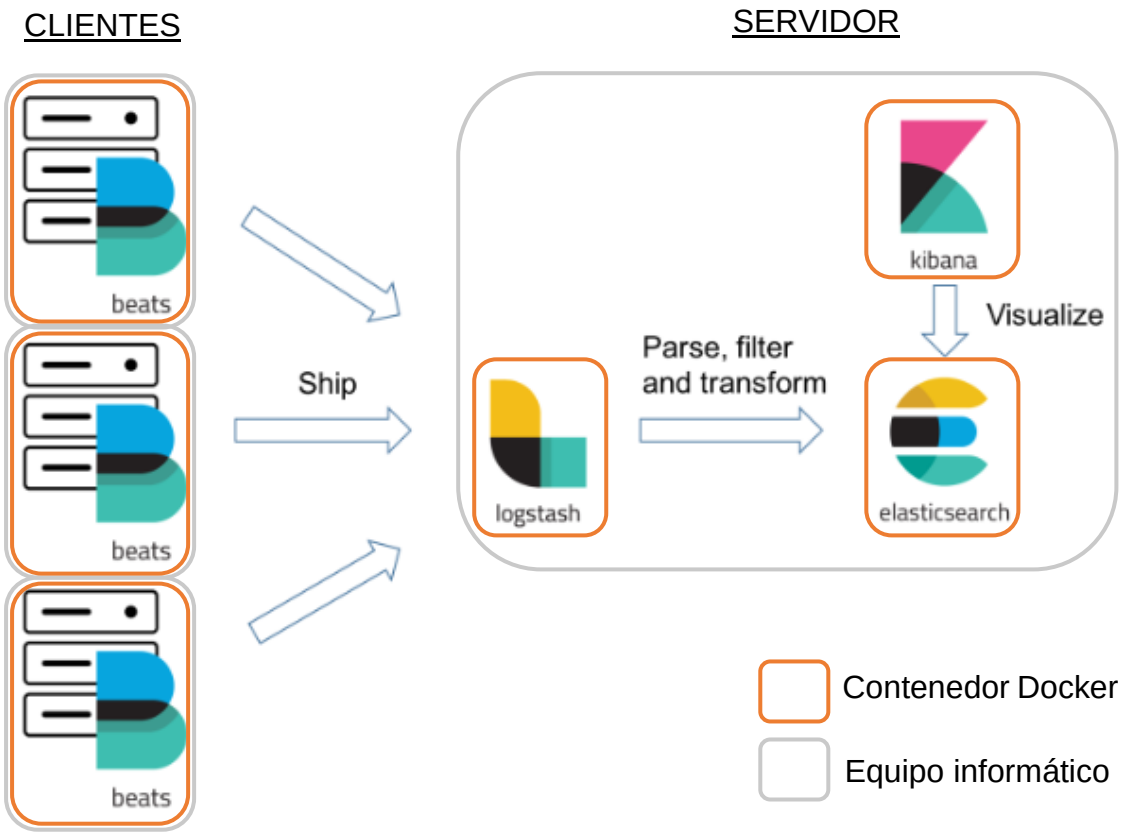


Ilustración 25. Arquitectura del sistema

5.3. CASOS DE USO

Un **caso de uso** es una descripción, desde el punto de vista del usuario, de los distintos pasos que debe realizar el sistema para completar un determinado proceso.

En este apartado se presentarán los casos de uso del sistema actual, utilizando como base la siguiente tabla:

CU-XX	
Nombre	Nombre del caso de uso CU-XX
Actores	Actores del caso de uso CU-XX
Objetivo	Objetivo del caso de uso CU-XX
Precondiciones	Precondiciones del caso de uso CU-XX
Postcondiciones	Postcondiciones del caso de uso CU-XX
Flujo normal	Flujo normal del caso de uso CU-XX
Flujo alternativo	Flujo alternativo del caso de uso CU-XX

Tabla 23. Plantilla de los casos de uso

Los atributos que se han utilizado para describir cada caso de uso son los siguientes:

- **Identificador:** código unívoco para cada caso de uso. Este está formado por:
    - **CU**: subcódigo que lo identifica como caso de uso.
    - **XX**: cifras numéricas que indican el número de caso de uso.
- **Nombre:** título representativo del caso de uso.
- **Actores:** aquellos roles o tipos de usuario que interactúan con el sistema en el caso de uso. En este sistema se encontrarán dos tipos de usuario:
    - **Administrador**: rol que posee los permisos para acceder a Kibana y, por tanto, a la información recolectada, y generar gráficas a partir de los datos que desee.
    - **Usuario**: rol que representa a un usuario distinto del administrador. En el sistema actual este rol no aparecerá sólo en los casos de uso, ya que el usuario no tiene contacto ni accede al sistema desarrollado en ningún momento.
- **Objetivo**: finalidad del caso de uso.

- **Precondiciones**: conjunto de condiciones previas que deben cumplirse para que comience el caso de uso.
- **Postcondiciones**: conjunto de condiciones que representan el estado del sistema tras la ejecución del caso de uso.
- **Flujo normal**: enumeración de los pasos o acciones a realizar en el escenario ideal del caso de uso, es decir, sin la aparición de problemas.
- **Flujo alternativo**: pasos o acciones que impiden que el caso de uso continúe su flujo normal y, en algunos casos, cause su finalización.

Cabe destacarse que, en el sistema actual, no son muchos los casos de uso, ya que la mayor parte del sistema funciona de forma automática sin que se tenga que realizar ninguna acción. Por lo tanto, únicamente se pueden incluir casos de uso relacionados las acciones que se pueden realizar o la información visible en la interfaz gráfica de Kibana.

A continuación, se definirán los casos de uso del sistema y, posteriormente, se creará una matriz de trazabilidad que relacione los casos de uso y los requisitos funcionales.

CU-01	
Nombre	Visualización de la información
Actores	Administrador
Objetivo	Observar información recibida en Kibana.
Precondiciones	El equipo cliente y servidor deben estar disponibles y con las herramientas correspondientes desplegadas.
Postcondiciones	Aparece información reciente en Kibana, o simplemente aparece información si es la primera vez que se ejecuta.
Flujo normal	1. El administrador accede a la interfaz web de Kibana a través de la URL http://SERVER_IP:5601, donde *SERVER_IP* se corresponde con la dirección IP del servidor. 2. El administrador introduce las credenciales para acceder a Kibana a través de Search Guard. 3. El administrador accede a la sección *Discover* en la barra de navegación de Kibana. 4. El administrador observa datos recibidos en los últimos minutos (o simplemente observa información si es la

	primera vez que lo ejecuta) gracias a la marca de tiempo asociada a cada línea de *log* recibida (*@timestamp)*
Flujo alternativo	2.1. Al acceder a la URL aparece un error. Se ha producido algún problema durante el despliegue de Kibana. El caso de uso finaliza. 3.1. El administrador no recuerda las credenciales de Kibana o el sistema de autenticación no funciona correctamente. El administrador no puede acceder a la interfaz web. El caso de uso finaliza. 4.1. El administrador no observa la recepción de nuevos datos. Se ha producido algún error de despliegue del resto de herramientas o en la conexión entre cliente y servidor. El caso de uso finaliza.

Tabla 24. CU-01

CU-02	
Nombre	Visualización de la información de varios equipos
Actores	Administrador
Objetivo	Observar información recibida de varios equipos en Kibana.
Precondiciones	El equipo cliente y servidor deben estar disponibles y con las herramientas correspondientes desplegadas.
Postcondiciones	Aparece información procedente de diferentes equipos en Kibana.
Flujo normal	*(Se repiten los pasos 1-3 del CU-01)* 4. El administrador observa datos recibidos de diferentes equipos gracias al campo que indica la dirección IP desde el que se recibió la línea de log (*beat.hostname*)
Flujo alternativo	*(Se repiten los pasos alternativos 2.1, 3.1 y 4.1 del CU-01)* 4.2. El administrador no observa la recepción de datos de diferentes ordenadores. Se ha producido algún error de despliegue de Filebeat en algunos de los equipos cliente o en su conexión con el servidor. El caso de uso finaliza.

Tabla 25. CU-02

CU-03	
Nombre	Recepción de nueva información
Actores	Administrador, usuario
Objetivo	Observar que las nuevas líneas de los *logs* se envían al servidor en cuanto son incorporadas al archivo correspondiente.
Precondiciones	- El equipo cliente y servidor deben estar disponibles y con las herramientas correspondientes desplegadas. - Se monitoriza el *log* "*auth.log*", por lo que este archivo se envía desde el equipo cliente al servidor.
Postcondiciones	Aparece nueva información en Kibana mientras se está presente en su interfaz gráfica.
Flujo normal	*(Se repiten los pasos 1-3 del CU-01)* 4. El administrador observa que hay datos recibidos en los últimos minutos. 5. Un usuario inicia sesión en un equipo cliente que se está monitorizando. 6. Tras unos segundos el administrador actualiza la página donde se encuentra Kibana o espera a que se actualiza automáticamente (si la opción se encuentra activada). 7. El administrador observa el inicio de sesión del usuario mediante la llegada de nuevas líneas de información a Kibana.
Flujo alternativo	*(Se repiten los pasos alternativos 2.1, 3.1 y 4.1 del CU-01)* 7.1. El administrador no observa la recepción de nuevos datos. Se ha producido algún error de despliegue de Filebeat en algunos de los equipos cliente o en su conexión con el servidor. El caso de uso finaliza.

Tabla 26. CU-03

CU-04	
Nombre	Consulta de información
Actores	Administrador
Objetivo	Realizar alguna consulta sobre la información almacenada en Elasticsearch a través de Kibana y obtener un resultado.
Precondiciones	El equipo cliente y servidor deben estar disponibles y con las herramientas correspondientes desplegadas.
Postcondiciones	Aparece un resultado de la consulta realizada.
Flujo normal	*(Se repiten los pasos 1-3 del CU-01)* 4. El administrador realiza una consulta a los índices de Elasticsearch utilizando la barra de búsqueda de la parte superior de la interfaz. 5. El administrador recibe el resultado de su consulta.
Flujo alternativo	*(Se repiten los pasos alternativos 2.1 y 3.1 del CU-01)* 5.1. El administrador no recibe resultado de su consulta o recibe un error. La conexión Elasticsearch-Kibana no se ha realizado correctamente o Elasticsearch no se ha desplegado. El caso de uso finaliza.

Tabla 27. CU-04

CU-05	
Nombre	Monitorización del sistema de monitorización
Actores	Administrador
Objetivo	Observar la monitorización en tiempo real de los componentes del *ELK Stack* (Elasticsearch, Logstash y Kibana).
Precondiciones	El equipo cliente y servidor deben estar disponibles y con las herramientas correspondientes desplegadas.
Postcondiciones	La monitorización del *ELK Stack* es visible a través de Kibana.
Flujo normal	*(Se repiten los pasos 1-2 del CU-01)*

	3. El administrador accede a la sección *Monitoring* en la barra de navegación de Kibana. 4. El administrador observa la monitorización en tiempo real de Elasticsearch, Logstash y Kibana.
Flujo alternativo	*(Se repiten los pasos alternativos 2.1 y 3.1 del CU-01)* 4.1. La monitorización no está disponible. El plugin de X-Pack no se ha instalado correctamente en alguno de los tres componentes. El caso de uso finaliza.

Tabla 28. CU-05

5.3.1. Matriz de trazabilidad

	RF-01	RF-02	RF-03	RF-04	RF-05	RF-06	RF-07	RF-08	RF-09	RF-10
CU-01	X	X			X	X	X		X	
CU-02	X	X	X		X	X	X		X	
CU-03	X	X		X	X	X	X		X	
CU-04								X		
CU-05										X

Tabla 29. Matriz de trazabilidad requisitos - casos de uso

6. IMPLEMENTACIÓN E IMPLANTACIÓN

Mientras que en el capítulo anterior se discutieron las herramientas a utilizar y se presentaron aquellas elegidas, en este capítulo se mencionará la versión de cada una de estas herramientas y del resto de programas o herramientas necesarios en el equipo para el correcto funcionamiento del sistema desarrollado.

Además, en este capítulo también se explicará el proceso de despliegue del sistema, lo que incluye tanto la instalación como la configuración de cada una de las herramientas elegidas.

6.1. TECNOLOGÍAS NECESARIAS

El sistema desarrollado podría funcionar en cualquier sistema operativo, pero el proceso de despliegue del sistema sería diferente, principalmente en la instalación de algunas herramientas.

Es por esto por lo que se especificarán las versiones de las herramientas o programas usados en el proceso de desarrollo del sistema:

- Sistema operativo: Debian 9.1, para simular el sistema operativo de determinadas aulas informáticas de la universidad.
- Navegador: Google Chrome, versión 67.0.X. De forma alternativa puede utilizarse cualquier otro navegador, ya que únicamente se necesita en el despliegue del sistema para la descarga de algunos componentes y para acceder a Kibana una vez el sistema está en funcionamiento.
- Docker: Docker CE versión 18.03.1 y Docker Compose versión 1.21.2, que coinciden con las últimas versiones disponibles en el día 15 de Abril de 2018.
- Herramientas de Elastic: Logstash, Elasticsearch, Kibana, Filebeat y X-Pack en su versión 6.23. Esta versión es la última disponible de estas herramientas en el día 15 de Abril de 2018.
- Search Guard: Search Guard versión 6.23-22.1, que es la versión compatible con la versión de las herramientas de Elastic (todas en la versión 6.23).

6.2. DESPLIEGUE DEL SISTEMA

6.2.1. Instalación y configuración de las herramientas

Como el proceso de instalación y configuración de cada una de las herramientas es un proceso que incluye una gran cantidad de pasos, se ha decidido definir este proceso a través de manuales incluidos como apéndices al final del actual documento. En estos manuales aparecen todos los pasos necesarios para obtener una correcta conexión de los componentes que forman el sistema desarrollado.

Por ello, mientras que en los apéndices se encuentra la información detallada del proceso de despliegue, en este apartado únicamente se mencionarán los pasos de forma general y haciendo referencia al apéndice donde se encuentran explicados detalladamente.

Siguiendo el procedimiento de capítulos anteriores, se explicarán los pasos para desplegar el equipo cliente y el/los equipo(s) cliente de forma separada:

- **Equipo servidor**:
    - Instalar Docker y Docker Compose

Véase el

        - APÉNDICE I: MANUAL DE INSTALACIÓN DE DOCKER Y DOCKER COMPOSE*.*
    - Instalar y configurar el ELK Stack (Elasticsearch, Logstash y Kibana)
        - Para facilitar la configuración y conexión entre las tres herramientas, se utilizará como base un repositorio de GitHub.
        - *Véase el* APÉNDICE II: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE LOGSTASH, ELASTICSEARCH Y KIBANA*.*
    - Instalar y configurar X-Pack.
        - Deben configurarse los archivos de configuración de cada uno de los componentes del ELK Stack para descargar e incluir el plugin.
        - Al no tener una licencia de pago, es necesario también especificar la desactivación del servicio de seguridad en los archivos de configuración.
        - *Véase el* APÉNDICE IV: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE X-PACK*.*
    - Instalar y configurar Search Guard.
        - Incluye la modificación de los archivos de configuración de cada una de las herramientas del ELK Stack para incluir la descarga y configuración de Search Guard y la generación de certificados para realizar el cifrado SSL/TLS que ofrece.

        - También se deben especificar las credenciales necesarias en cada herramienta por la activación del control de acceso incluido en la instalación del plugin.
        - *Véase el* APÉNDICE V: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE SEARCH GUARD*.*
    - Obtener e instalar una licencia *Basic* en Elastic
        - Esta licencia es necesaria para, principalmente, disfrutar de los servicios gratuitos de X-Pack (sobre todo la monitorización).
        - *Véase el* APÉNDICE VI: MANUAL DE OBTENCIÓN E INSTALACIÓN DE UNA LICENCIA TIPO *BASIC* EN ELASTIC*.*

- **Equipo(s) cliente**:
    - Instalar Docker y Docker Compose.

Véase el

        - APÉNDICE I: MANUAL DE INSTALACIÓN DE DOCKER Y DOCKER COMPOSE*.*
    - Instalar y configurar Filebeat.
        - Además, se configurará Logstash en el equipo servidor para conectar ambas herramientas y, por ende, el cliente y el servidor.
        - *Véase el* APÉNDICE III: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE FILEBEAT. CONEXIÓN ENTRE FILEBEAT Y LOGSTASH*.*

7. EVALUACIÓN

7.1. CASOS DE PRUEBA

Un caso de prueba es un conjunto de condiciones o pasos que demuestra que el sistema funciona de forma satisfactoria, es decir, que ofrece los resultados esperados.

El diseño que se seguirá para definir cada caso de prueba se mostrará en la siguiente tabla:

PRU-XX	
Nombre	Nombre del caso de prueba PRU-XX
Objetivo	Objetivo del caso de prueba PRU-XX
Caso de uso	Identificador del caso de uso del caso de prueba PRU-XX
Procedimiento	Procedimiento del caso de prueba PRU-XX
Resultado obtenido	Resultado obtenido del caso de prueba PRU-XX

Tabla 30. Plantilla de los casos de prueba

Los atributos que se han utilizado para describir cada caso de prueba son los siguientes:

- **Identificador:** código unívoco para cada caso de prueba. Este está formado por:
  - **CU**: subcódigo que lo identifica como caso de prueba.
  - **XX**: cifras numéricas que indican el número de caso de prueba.
- **Nombre:** título del caso de prueba.
- **Objetivo:** funcionalidad o comportamiento que quiere evaluar el caso de prueba.
- **Caso de uso:** identificador del caso de uso que utiliza como base el caso de prueba.
- **Procedimiento:** pasos a realizar para la realización del caso de prueba.
- **Resultado obtenido:** resultado obtenido tras realizar el procedimiento del caso de prueba.

A continuación, se definirán los casos de prueba del sistema y, posteriormente, se creará una matriz de trazabilidad que relacione los casos de prueba y los casos de uso.

PRU-01	
Nombre	Visualización de la información
Objetivo	Comprobar que la información se transmite correctamente del equipo cliente al servidor.
Caso de uso	CU-01
Procedimiento	1. Se accede a la interfaz web de Kibana a través de la URL http://SERVER_IP:5601, donde *SERVER_IP* se corresponde con la dirección IP del servidor. 2. Se introducen las credenciales para acceder a Kibana a través de Search Guard. 3. Se accede a la sección *Discover* en la barra de navegación de Kibana.
Resultado obtenido	En la sección *Discover* de Kibana se observan datos recibidos en los últimos minutos. Si es la primera vez que se ejecuta el sistema, simplemente aparece información.

Tabla 31. PRU-01

PRU-02	
Nombre	Visualización de la información de varios equipos
Objetivo	Comprobar que la información se transmite correctamente de varios equipos cliente al servidor de forma simultánea.
Caso de uso	CU-02
Procedimiento	1. Se accede a la interfaz web de Kibana a través de la URL http://SERVER_IP:5601, donde *SERVER_IP* se corresponde con la dirección IP del servidor. 2. Se introducen las credenciales para acceder a Kibana a través de Search Guard. 3. Se accede a la sección *Discover* en la barra de navegación de Kibana.
Resultado obtenido	En la sección *Discover* de Kibana se observan datos recibidos en los últimos minutos procedentes de diferentes equipos cliente. Esto se puede observar a través del campo *beat.hostname*.

Tabla 32. PRU-02

PRU-03	
Nombre	Recepción de nueva información
Objetivo	Comprobar que la información se transmite constantemente del equipo cliente al servidor, de forma que cuando una nueva línea aparece en un *log* monitorizado, a los pocos segundos o minutos esta línea se hace visible en Kibana.
Caso de uso	CU-03
Procedimiento	1. Se accede a la interfaz web de Kibana a través de la URL http://SERVER_IP:5601, donde *SERVER_IP* se corresponde con la dirección IP del servidor. 2. Se introducen las credenciales para acceder a Kibana a través de Search Guard. 3. Se accede a la sección *Discover* en la barra de navegación de Kibana. 4. Se espera unos segundos/minutos y se actualiza la página o se espera la actualización automática de Kibana (si se encuentra activa).
Resultado obtenido	En la sección *Discover* de Kibana se observan nuevos datos recibidos frente a los presentes antes de actualizar la página.

Tabla 33. PRU-03

PRU-04	
Nombre	Consulta de información
Objetivo	Comprobar que se pueden realizar consultas a los índices de Elasticsearch a través de Kibana.
Caso de uso	CU-04
Procedimiento	1. Se accede a la interfaz web de Kibana a través de la URL http://SERVER_IP:5601, donde *SERVER_IP* se corresponde con la dirección IP del servidor. 2. Se introducen las credenciales para acceder a Kibana a través de Search Guard. 3. Se accede a la sección *Discover* en la barra de navegación de Kibana.

	4. Se realiza una consulta los índices de Elasticsearch haciendo uso del cuadro de búsqueda situado en la parte superior de la pantalla.
Resultado obtenido	Se recibe un resultado coherente, distinto de error, con la consulta realizada.

Tabla 34. PRU-04

PRU-05	
Nombre	Monitorización del sistema de monitorización
Objetivo	Comprobar que la monitorización del sistema de monitorización base (Logstash, Elasticsearch y Kibana) se encuentra activa tras el despliegue del sistema.
Caso de uso	CU-05
Procedimiento	1. Se accede a la interfaz web de Kibana a través de la URL http://SERVER_IP:5601, donde *SERVER_IP* se corresponde con la dirección IP del servidor. 2. Se introducen las credenciales para acceder a Kibana a través de Search Guard. 3. Se accede a la sección *Monitoring* en la barra de navegación de Kibana.
Resultado obtenido	En la sección *Monitoring* de Kibana se observa un apartado para cada una de las herramientas del sistema base (Logstash, Elasticsearch y Kibana) y se observa que el estado o salud de cada una de las herramientas se encuentra en verde o, al menos, en amarillo.

Tabla 35. PRU-05

7.1.1. Matriz de trazabilidad

	CU-01	CU-02	CU-03	CU-04	CU-05
PRU-01	X				
PRU-02		X			
PRU-03			X		
PRU-04				X	
PRU-05					X

Tabla 36. Matriz de trazabilidad casos de uso - casos de prueba

8. PLANIFICACIÓN Y PRESUPUESTO

8.1. PLANIFICACIÓN

Para comenzar a realizar la planificación del trabajo, era necesario conocer en qué etapas y tareas se iba a dividir el mismo. Así, las tareas que se consideraron necesarias para completar el trabajo fueron las siguientes:

- Análisis del estado de la cuestión:
  - Estudio de la situación actual del análisis de logs.
  - Estudio de las principales herramientas existentes para el análisis de logs: Sumo Logic, Splunk y ELK Stack.
  - Elección del sistema base a utilizar en el trabajo: ELK Stack.
- Análisis del sistema a desarrollar:
  - Aprendizaje general de herramientas posibles
  - Redacción de los requisitos funcionales y no funcionales.
  - Elección y aprendizaje asociado a Docker.
  - Aprendizaje asociado al ELK Stack (Elasticsearch, Logstash y Kibana) y Filebeat.
  - Aprendizaje asociado a X-Pack y Search Guard (plugins del ELK Stack)
- Diseño de la solución:
  - Diseño de la arquitectura del sistema a desarrollar.
  - Diseño de los gráficos y visualizaciones a usar en Kibana.
- Implementación e implantación de la solución:
  - Instalación de Docker y Docker Compose.
  - Instalación y configuración del ELK Stack (Logstash, Elasticsearch y Kibana) y Filebeat.
  - Instalación y configuración de X-Pack y Search Guard.
- Evaluación
  - Realización de pruebas en un entorno doméstico.
  - Realización de pruebas en un equipo de pruebas del laboratorio.
- Redacción de la memoria

8.1.1. Planificación estimada

Siguiendo las etapas definidas previamente, se decidió elaborar una tabla en la que se indicara, para cada etapa, un valor de esfuerzo (en una escala del 1 al 5), según el tiempo previsto para la realización del objetivo de la etapa.

Etapa	Esfuerzo (del 1 al 5)
Análisis del estado de la cuestión	
Estudio de la situación actual del análisis de logs	1
Estudio de las principales herramientas existentes para el análisis de logs: Sumo Logic, Splunk y ELK Stack	2
Elección del sistema a utilizar en el trabajo: ELK Stack	1
Análisis del sistema a desarrollar	
Aprendizaje general de herramientas posibles	2
Redacción de los requisitos funcionales y no funcionales	1
Elección y aprendizaje asociado a Docker	2
Aprendizaje asociado al ELK Stack (Elasticsearch, Logstash y Kibana) y Filebeat	3
Aprendizaje asociado a X-Pack y Search Guard	2
Diseño de la solución	
Diseño de la arquitectura del sistema a desarrollar	2
Diseño de los gráficos y visualizaciones a usar en Kibana	1
Implementación e implantación de la solución	
Instalación de Docker y Docker Compose	3

Etapa	Esfuerzo (del 1 al 5)
Instalación y configuración del ELK Stack (Logstash, Elasticsearch y Kibana) y Filebeat	5
Instalación y configuración de X-Pack y Search Guard	4
Evaluación de la solución	
Realización de pruebas en un entorno doméstico	2
Realización de pruebas en un equipo de pruebas del laboratorio	3
Redacción de la memoria	5

Tabla 37. Planificación estimada (esfuerzos)

Según la tabla anterior, se decidió que cada nivel de esfuerzo implicado en una actividad implicaba un periodo de 5 días para su realización, a excepción de la redacción de la memoria, que se realiza durante todo el proceso y requiere una semana más para terminar de redactarse y corregir errores.

De esta forma, conociendo la fecha de inicio del proyecto (1 de noviembre de 2017) y el esfuerzo considerado para cada etapa, pudo realizarse la siguiente planificación estimada.

Etapa	Fecha de inicio	Fecha de fin
Análisis del estado de la cuestión	**01/11/2017**	**20/11/2017**
Estudio de la situación actual del análisis de logs	01/11/2017	05/11/2017
Estudio de las principales herramientas existentes para el análisis de logs: Sumo Logic, Splunk y ELK Stack	06/11/2017	15/11/2017
Elección del sistema a utilizar en el trabajo: ELK Stack	16/11/2017	20/11/2017
Análisis del sistema a desarrollar	**21/11/2017**	**09/01/2018**

Etapa	Fecha de inicio	Fecha de fin
Aprendizaje general de herramientas posibles	21/11/2017	30/11/2017
Redacción de los requisitos funcionales y no funcionales	01/12/2017	05/12/2017
Elección y aprendizaje asociado a Docker	06/12/2017	15/12/2017
Aprendizaje asociado al ELK Stack (Elasticsearch, Logstash y Kibana) y Filebeat	16/12/2017	30/12/2017
Aprendizaje asociado a X-Pack y Search Guard	31/12/2017	09/01/2018
Diseño de la solución	**10/01/2018**	**24/01/2018**
Diseño de la arquitectura del sistema a desarrollar	10/01/2018	19/01/2018
Diseño de los gráficos y visualizaciones a usar en Kibana	20/01/2018	24/01/2018
Implementación e implantación de la solución	**25/01/2018**	**25/03/2018**
Instalación de Docker y Docker Compose	25/01/2018	08/02/2018
Instalación y configuración del ELK Stack (Logstash, Elasticsearch y Kibana) y Filebeat	09/02/2018	05/03/2018
Instalación y configuración de X-Pack y Search Guard	06/03/2018	25/03/2018
Evaluación de la solución	**26/03/2018**	**19/04/2018**
Realización de pruebas en un entorno doméstico	26/03/2018	04/04/2018
Realización de pruebas en un equipo de pruebas del laboratorio	05/03/2018	19/04/2018
Redacción de la memoria	**01/11/2017**	**26/04/2018**

Tabla 38. Planificación estimada

De esta forma, el tiempo estimado la realización del proyecto sería desde el 1 de Noviembre de 2017 al 26 de Abril de 2018.

8.1.1.1. Diagrama de Gantt

Siguiendo la planificación estimada anterior, se realizó el diagrama de Gantt correspondiente.

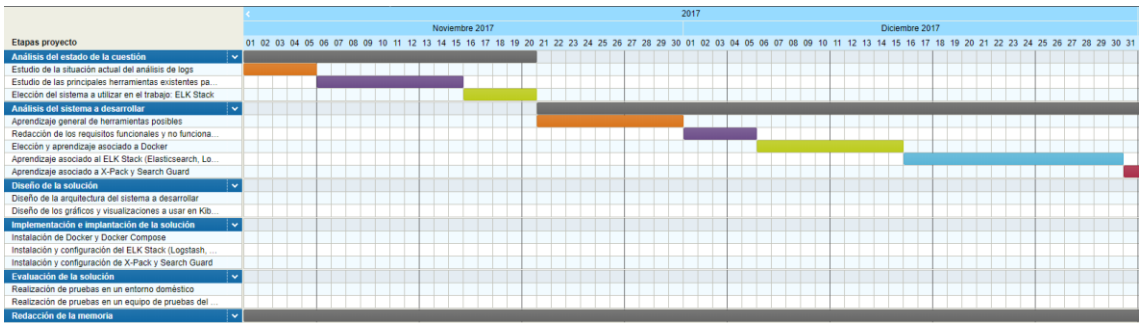


Tabla 39. Diagrama de Gantt estimado 1

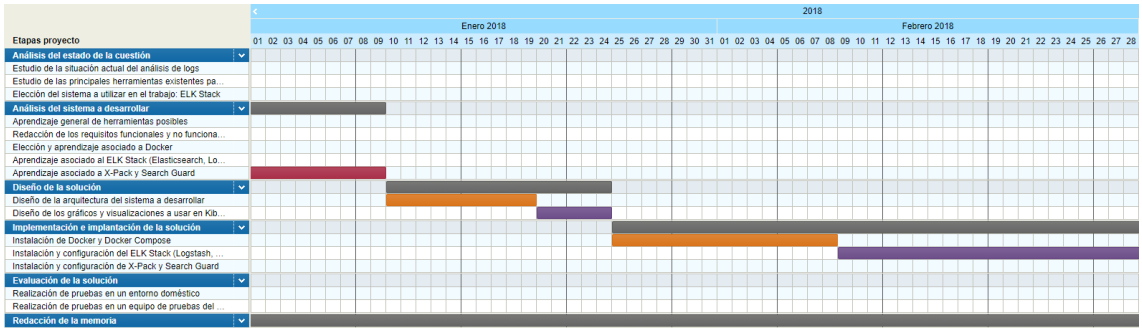


Tabla 40. Diagrama de Gantt estimado 2

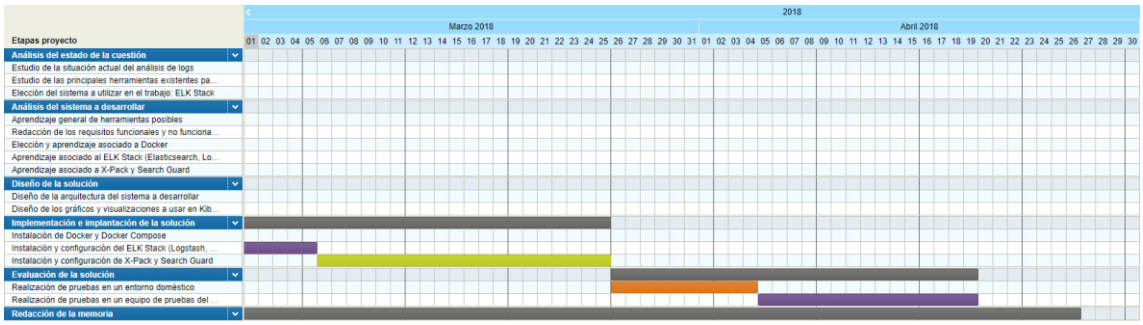


Tabla 41. Diagrama de Gantt estimado 3

8.1.2. Planificación real

Debido a diversos factores externos al proyecto (festivos, resto de asignaturas del curso académico, etc) así como relacionados con el mismo (aparición de imprevistos,

etapas que requerían más/menos tiempo del estimado, etc), las etapas del proyecto no se han cumplido, en la mayoría de los casos, en las fechas estimadas.

Es por esto que es necesario incluir una tabla donde se muestre la planificación final o real, que es la que refleja los tiempos reales que han sido necesarios para alcanzar el objetivo de cada etapa.

Etapa	Fecha de inicio	Fecha de fin
Análisis del estado de la cuestión	**01/11/2017**	**13/11/2017**
Estudio de la situación actual del análisis de logs	01/11/2017	05/11/2017
Estudio de las principales herramientas existentes para el análisis de logs: Sumo Logic, Splunk y ELK Stack	06/11/2017	11/11/2017
Elección del sistema a utilizar en el trabajo: ELK Stack	12/11/2017	13/11/2017
Análisis del sistema a desarrollar	**14/11/2017**	**07/01/2018**
Aprendizaje general de herramientas posibles	14/11/2017	20/11/2017
Redacción de los requisitos funcionales y no funcionales	21/11/2017	25/11/2017
Elección y aprendizaje asociado a Docker	26/11/2017	07/12/2017
Aprendizaje asociado al ELK Stack (Elasticsearch, Logstash y Kibana) y Filebeat	08/12/2017	27/12/2017
Aprendizaje asociado a X-Pack y Search Guard	28/12/2017	07/01/2018
Diseño de la solución	**08/01/2018**	**17/01/2018**
Diseño de la arquitectura del sistema a desarrollar	08/01/2018	14/01/2018
Diseño de los gráficos y visualizaciones a usar en Kibana	15/01/2018	17/01/2018
Implementación e implantación de la solución	**18/01/2018**	**30/03/2018**

Etapa	Fecha de inicio	Fecha de fin
Instalación de Docker y Docker Compose	18/01/2018	27/01/2018
Instalación y configuración del ELK Stack (Logstash, Elasticsearch y Kibana) y Filebeat	28/01/2018	28/02/2018
Instalación y configuración de X-Pack y Search Guard	01/03/2018	30/03/2018
Evaluación de la solución	**31/03/2018**	**23/04/2018**
Realización de pruebas en un entorno doméstico	31/03/2018	09/04/2018
Realización de pruebas en un equipo de pruebas del laboratorio	10/03/2018	23/04/2018
Redacción de la memoria	**01/11/2017**	**07/05/2018**

Tabla 42. Planificación real

De esta forma, la fecha final real del proyecto fue el **7 de mayo de 2018**.

8.1.2.1. Diagrama de Gantt

De la misma forma que en la planificación estimada, en este apartado se añadirá el diagrama de Gantt correspondiente a la planificación real.

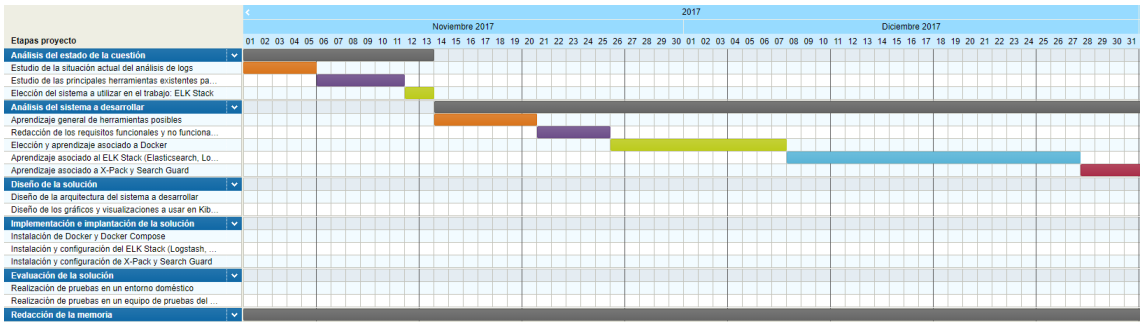


Tabla 43. Diagrama de Gantt real 1

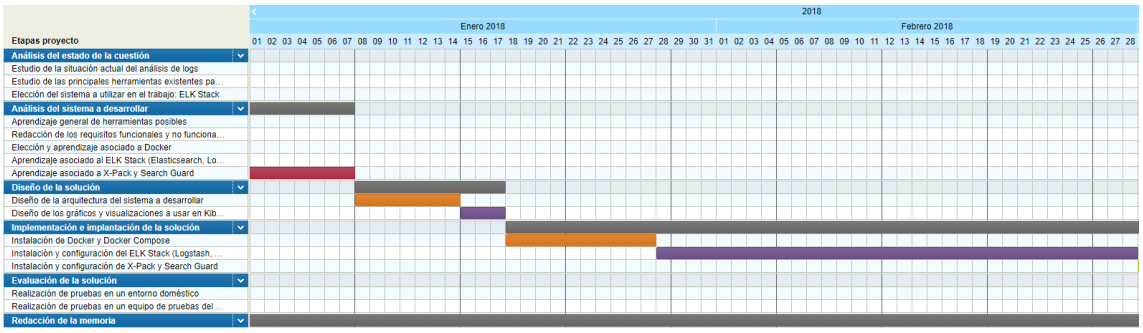


Tabla 44. Diagrama de Gantt real 2

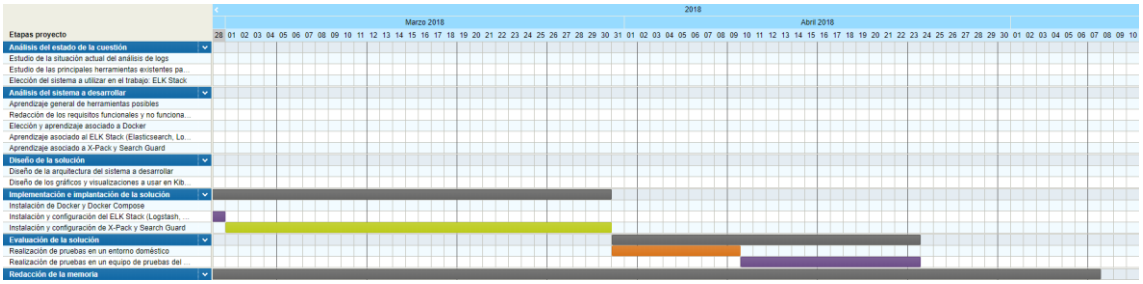


Tabla 45. Diagrama de Gantt real 3

8.2. PRESUPUESTO

8.2.1. Gasto de personal

Para el desarrollo del sistema, serán necesarias cuatro personas para desempeñar cada uno de los siguientes roles:

- **Jefe de proyecto**: encargado de coordinar, dirigir y gestionar al resto del equipo durante la realización del proyecto. Además, se encargará de tomar las decisiones más críticas que surjan durante la etapa de desarrollo.
- **Analista/Diseñador**: encargado de la extracción de los distintos requisitos del sistema, así como del desarrollo del sistema desde el punto de vista del diseño.
- **Responsable de implantación**: encargado de instalar y configurar cada una de las herramientas necesarias para el despliegue del sistema diseñado.
- **Responsable de pruebas**: encargado de la realización de las pruebas necesarias para comprobar el correcto funcionamiento del sistema y así evitar comportamientos inesperados.

A continuación, se detallará en una tabla los sueldos asociados a cada uno de los anteriores roles, según el portal de empleo Infojobs. Además, para los cálculos referidos a este gasto de personal, se tendrá en cuenta un aumento del 23,6% para la cotización de la Seguridad Social, según el Régimen General de la Seguridad Social.

Rol	Sueldo bruto / año	Seguridad Social	Gasto final / año
Jefe de proyecto	24.000,00 €	5.644,00 €	29.644,00 €
Analista / Diseñador	30.000,00 €	7.080,00 €	37.080,00 €
Responsable de implantación	28.000,00 €	6.608,00 €	34.608,00 €
Responsable de pruebas	32.000,00 €	7.552,00 €	39.552,00 €
		TOTAL (I.V.A. incluido)	**140.884,00 €**
		TOTAL 7 MESES (I.V.A. incluido)	**82.182,00 €**

Tabla 46. Gastos de personal

Según los resultados de la tabla anterior, el total de gasto de personal por año sería de 140.884,00 €, pero, como el proyecto tiene una duración de 7 meses (en vez de un año), el gasto de personal del proyecto sería **82.182,00 €** (I.V.A. incluido).

Antes de terminar el actual apartado merece realizarse la siguiente aclaración: el valor anteriormente calculado se corresponde con los gastos de personal de una determinada empresa si su objetivo fuera la realización completa del proyecto descrito en este documento. Por otro lado, en el proceso real de desarrollo del sistema que ha durado esos 7 meses (y pido permiso para utilizar la primera persona únicamente en este párrafo), he sido únicamente yo el encargado de cubrir todos los roles, por lo que el personal estaba formado por una única persona, sin sueldo asociado.

8.2.2. Gastos de material

Dentro de este apartado de gastos se encuentran los correspondientes al material, el cual, en este trabajo, está formado principalmente por los equipos necesarios para el despliegue del sistema desarrollado y las licencias necesarias para el funcionamiento de cada una de las herramientas, además del alquiler de un local donde poder desarrollar el sistema.

En primer lugar, para el equipo servidor, donde se realizará la monitorización y donde se encuentra la base del sistema (Logstash, Elasticsearch y Kibana), será necesario un equipo informático, cuyo valor promedio es de 650 euros (I.V.A. incluido) según Context, la empresa europea de análisis de mercado de IT [REFERENCIA]. Podría ser necesaria la ampliación del disco duro del sistema para aumentar la cantidad

de información que puede almacenarse en Elasticsearch, pero eso no se tendrá en cuenta de cara al presupuesto.

Con respecto a los equipos clientes, no será necesaria la compra de ningún equipo para desempeñar esta función. Esto se debe a que el actual trabajo está destinado a la monitorización de equipos existentes y en uso, y esto no requiere la compra de nuevos equipos.

Por otro lado, con respecto a las licencias utilizadas, todos los programas son gratuitos o se usan en su versión o licencia gratuita, por lo que no hay que valorar gastos en cuanto a lo que a licencias se refiere.

Por último, con respecto al lugar donde se desarrollará el proyecto, se alquilará una oficina amueblada durante los 7 meses del proyecto. Mirando en la web de FotoCasa, el precio de una oficina amueblada con suficiente espacio para 4 personas y sus correspondientes ordenadores es de, aproximadamente, 400 euros al mes.

Por lo tanto, y teniendo en cuenta lo explicado en este apartado, los gastos de material quedarían de la siguiente forma:

Producto	Precio
1 x Ordenador promedio	650,00 €
1 x Alquiler oficina amueblada (un mes)	400,00 €
TOTAL (I.V.A. incluido)	**650,00 €**
TOTAL 7 MESES (I.V.A. incluido)	**3.450,00 €**

Tabla 47. Gastos de material

De esta forma, el gasto de material final sería de **3.450,00 €** (I.V.A. incluido).

8.2.3. Otros gastos directos

Además del gasto de personal y de material, otros gastos directos a tener en cuenta en el presupuesto final del sistema serían la luz y el ADSL, necesarios para el funcionamiento de los equipos y el correcto despliegue y funcionamiento del sistema desarrollado, respectivamente.

En la mayoría de los proyectos, teniendo como referencia lo mencionado [REFERENCIA], tanto la luz como el ADSL se considerarían gastos indirectos de cara al presupuesto, ya que no se encuentran directamente asociados con el producto en sí. Sin embargo, en el proyecto actual, ambos servicios son totalmente imprescindibles

para la instalación, despliegue y mantenimiento del sistema desarrollado, y es esta la principal razón por la que se han considerado como gastos directos.

Actualmente, teniendo como referencia el servicio proporcionado por Gas Natural Fenosa, el precio del kilovatio por hora es 0,1480€/kWh (I.V.A. incluido). Teniendo en cuenta este valor, un consumo de potencia promedio de 200 W (por hora) y una necesidad del servicio de 24h todos los días de la semana, el gasto de la luz sería el siguiente:

Servicio	Consumo de potencia medio	Tiempo de uso	Precio del kwH	Precio/mes
Consumo de luz	200 W	24 horas, 7 días/semana	0,1480€/kWh	21,31 €
TOTAL (I.V.A. incluido)				**21,31 €**

Tabla 48. Gasto de la luz

Con respecto al ADSL, Jazztel ofrece una fibra óptica simétrica de 50 Mb por un precio de 43,95€/mes (I.V.A. incluido). Como la compañía en sí es irrelevante (más allá de un buen soporte y servicio), se elegirá Jazztel como compañía para el suministro de ADSL y, por tanto, ese es el gasto que se considerará. De esta forma, el gasto del ADSL sería:

Servicio	Precio/mes
Fibra óptica simétrica de 50 Mb	43,95 €
TOTAL (I.V.A. incluido)	**43,95 €**

Tabla 49. Gasto de ADSL

Finalmente, el cálculo final de los gastos directos para los 7 meses de duración del proyecto sería el presentado en la siguiente tabla:

Servicio	Precio/mes
Consumo de luz (Gas Natural Fenosa)	21,31 €
Fibra óptica simétrica de 50 Mb (Jazztel)	43,95 €
TOTAL (I.V.A. incluido)	**65,26 €**
TOTAL 7 MESES (I.V.A. incluido)	**456,82 €**

Tabla 50. Otros gastos indirectos

Por lo tanto, el gasto de la luz y del ADSL para la duración del proyecto (7 meses) sería de **456,82 €** (I.V.A. incluido).

En la siguiente tabla se podrán encontrar los enlaces donde se ha buscado la información correspondiente a los precios de los servicios incluidos en este apartado. Se ha decidido no incluirlos directamente en la bibliografía por no considerarse información relevante de cara al desarrollo del sistema y la escritura de la memoria.

Servicio	Compañía	URL
Luz	Gas Natural Fenosa	https://www.gasnaturalfenosa.es/hogar/luz_o_gas/contratar_luz_o_gas/tarifas_luz_y_gas/tarifas_estables
ADSL	Jazztel	https://www.jazztel.com/internet/ofertas-fibra.html

Tabla 51. Información de gastos indirectos

8.2.4. Presupuesto final

Siguiendo los pasos de los anteriores apartados, el presupuesto final del proyecto para la duración del mismo (7 meses) estaría formado por los gastos de personal, los gastos de material y otros gastos directos (luz y ADSL). Además, a este resultado final hay que sumarle un 10% del mismo, destinado a los riesgos.

Así, la estructura del presupuesto final quedaría de la siguiente forma:

Gastos	Precio (7 meses)
Gastos de personal	82.182,00 €
Gastos de material	3.450,00 €
Otros gastos directos	456,82 €
TOTAL sin riesgos (I.V.A. incluido)	**86.088,82 €**
Riesgos (10%)	8.608,88 €
TOTAL (I.V.A. incluido)	**94.697,70 €**

Tabla 52. Presupuesto final

Por lo tanto, el precio total del proyecto durante los 7 meses de duración del mismo sería de **94.697,70 €** (I.V.A. incluido).

9. CONCLUSIONES Y TRABAJOS FUTUROS

9.1. CONCLUSIONES

9.1.1. Producto

El sistema desarrollado en este proyecto, aunque en un inicio estaba formado únicamente por los componentes del *ELK Stack*, ha acabado estando formado por un gran número de herramientas: Filebeat, Logstash, Elasticsearch, Kibana, X-Pack y Search Guard.

Gracias a estas herramientas, se ha podido realizar con éxito el principal objetivo del proyecto: la creación de un sistema gratuito de monitorización en tiempo real de *logs* que incluya el envío, el almacenamiento y la visualización de los mismos. Además, la inclusión de X-Pack en las herramientas que forman el sistema ha permitido la monitorización del propio sistema de monitorización desarrollado, extendiendo al propio sistema la monitorización buscada en el proyecto.

Con respecto al segundo objetivo, el empleo de mecanismos de seguridad, aunque se ha implementado un control de acceso básico y un cifrado del clúster de Elasticsearch a través de Search Guard, la constante aparición de errores y el desconocimiento de su solución ha impedido el establecimiento de los mecanismos de seguridad que se desearon en un inicio. Esto se explicará con mayor detalle en el siguiente apartado.

9.1.2. Proceso

Durante el proceso de desarrollo del sistema, son varios los problemas que han surgido y que han ralentizado, incluso detenido, el normal desarrollo del proyecto. Dentro de estos problemas, los que merecen destacarse son los dos siguientes:

- **Problemas en la implementación y despliegue del sistema**: sin duda alguna, la configuración de las distintas herramientas y su interconexión ha sido la principal razón que ha ralentizado el proyecto. La configuración de cada herramienta incluye un gran conjunto de pares parámetro-valor que deben estar correctos para que la herramienta llegue a desplegarse. Una vez desplegada cada herramienta, la interconexión entre ellas para que la información pasara del equipo cliente al equipo servidor dependía de la coincidencia de la información presente en los archivos de configuración de las herramientas implicadas, lo que se trataba de una tarea aún más tediosa.
- **Problemas con el uso de certificados**: Search Guard ofrece una gran cantidad de opciones para el desarrollo de los certificados usados en el cifrado SSL/TLS que ofrece, y muchos de ellos son certificados de ejemplo o certificados generados por herramientas que Search Guard ha desarrollado para su creación. Sin embargo, aun usando cada uno de estos sistemas de generación de certificados, ha sido imposible la validación de certificados cuando una

herramienta debía validar el certificado de otra. También se probó a crear los certificados usando *OpenSSL*, pero el resultado siguió siendo el mismo.

Este problema no sólo ha impedido el total despliegue de Search Guard para aumentar la seguridad de las interconexiones entre las herramientas, sino que también ha impedido el establecimiento de un cifrado SSL/TLS en las comunicaciones cliente-servidor a través de la configuración de Filebeat y Logstash.

Por otro lado, el conocimiento actual de las distintas herramientas que forman el sistema desarrollado permitiría que, si se realizara un proyecto que contara con estas herramientas, este se realizaría en un periodo de tiempo mucho menor, ya que no habría que invertir tanto tiempo en el aprendizaje de las herramientas.

De la misma forma, un número mayor de personas implicadas en el proyecto habría requerido menos tiempo para el desarrollo del sistema, ya que se podría haber dividido el aprendizaje y la configuración de las distintas herramientas que lo forman.

9.1.3. Personales

Son varias las asignaturas de la carrera realizada que han sido útiles en el desarrollo del sistema y en la redacción de la actual memoria. Sobre todo, merecen destacarse las siguientes asignaturas:

- "**Ingeniería del Software**" y "**Dirección de Proyectos de Desarrollo de Software**": estas asignaturas han facilitado la planificación de las etapas del proyecto y, sobre todo la redacción de la memoria. Esto se debe a la gran cantidad de documentos escritos en dichas asignaturas que se podían tomar como referencia para la estructuración de la memoria, además de los diferentes ejemplos aportados para definir las tablas de los requisitos, de los casos de uso o de los casos de prueba.
- "**Criptografía y Seguridad Informática**" e "**Ingeniería de la Seguridad**": estas asignaturas aportaron los principales conceptos de seguridad y de permisos sobre los ficheros, lo que se ha utilizado como base para utilizar correctamente todos los distintos tipos de archivos utilizados a lo largo del desarrollo de proyecto.
- "**Seguridad en Dispositivos Móviles**": esta asignatura fue la primera toma de contacto con el uso de certificados, lo que ha sido especialmente útil a la hora de configurar el cifrado SSL/TLS ofrecido por Search Guard.
- "**Redes de Ordenadores**": esta asignatura permitió la adquisición de conocimientos relacionados con la interconexión entre distintos equipos informáticos, lo que ha sido útil para la conexión entre cliente y servidor.

- "**Técnicas de Búsqueda y Uso de la Información**": útil para la redacción de la bibliografía, situada al final del actual documento.

Sin embargo, aunque son muchas las asignaturas que han servido de ayuda para la realización del proyecto, gran parte del sistema desarrollado contenía conceptos nuevos y el uso de programas que no se habían utilizado previamente. Esto incluye:

- **Contenedores y Docker**: este proyecto ha sido la primera toma de contacto con los contenedores de software, por lo que ha sido necesaria el aprendizaje de su funcionamiento y estructura, así como del programa que los crea y despliega, Docker.
- **Múltiples herramientas**: son muchas las herramientas incluidas en el sistema, y cada una de ellas tiene unas funcionalidades y configuraciones distintas, las cuáles han tenido que ser aprendidas para poder realizar su interconexión y configuración.
- **Uso de certificados**: aunque en la asignatura de "Seguridad en Dispositivos Móviles" se tuvo una primera toma de contacto con el uso de certificados, activar la seguridad en el sistema a través de Search Guard ha implicado un mayor aprendizaje de los mismos, ya que son muchos más los archivos necesarios y los comandos utilizados para la creación de estos archivos. De hecho, como se explicó previamente en este apartado de conclusiones, ha habido algunos aspectos de la seguridad que no se han podido implementar debido a la dificultad de autenticación de los certificados entre las herramientas.
- **Formato .yml**: la mayor parte de los archivos de configuración de los programas utilizados en el sistema desarrollado se encontraban en este formato, el cual apenas se ha utilizado durante la carrera y que, por tanto, ha implicado un aprendizaje del mismo para evitar errores en el despliegue.

9.2. TRABAJOS FUTUROS

Entre los principales trabajos futuros del sistema desarrollado se encuentran:

- **Mejorar la seguridad del sistema**: como bien se ha presentado en el anterior apartado de este capítulo, son muchos los errores que han surgido a la hora de implementar los mecanismos de seguridad existentes en el sistema desarrollado, principalmente para el cifrado de la interconexión entre las herramientas del *ELK Stack,* y entre el cliente y el servidor (Filebeat-Logstash). Por ello, como trabajo futuro se busca la habilitación completa de este cifrado SSL/TLS para hacer las comunicaciones del sistema mucho más seguras.

- **Ampliar los usos del sistema**: el sistema desarrollado es un sistema que funciona, es decir, realiza el objetivo deseado sin errores y de la forma deseada. Por ello, el sistema actual podría utilizarse no sólo para la monitorización de la información situada en cualquier *log* de los equipos cliente, sino que podría inyectarse en unos *logs* personalizado una determinada información que no se encuentre documentada en el propio sistema operativo y monitorizarla mediante el sistema desarrollado.

  De hecho, una de las opciones que se había visto posible y que sería uno de los siguientes casos a dar al sistema, era la concatenación en *logs* personalizados de información relacionada con el rendimiento de los equipos: temperatura de la CPU, ocupación de los discos duros, velocidad de los ventiladores, etc. De esta forma, se podría monitorizar el rendimiento de los distintos ordenadores de un aula informática desde un único ordenador y se podría detectar comportamientos anómalos debido a sobrecargas del sistema o a ataques de usuarios externos.

- **Mejorar la visualización de la información:** de la misma forma que Logstash ofrece un potente sistema de filtrado, Kibana ofrece una gran cantidad de tipos de gráficos y *dashboards* para la visualización de la información. Ambos son servicios que pueden dar lugar a un sinfín de oportunidades para mejorar tanto la cantidad de información que se presenta como la reunión de la misma en un único “pantallazo”.

APÉNDICE I: MANUAL DE INSTALACIÓN DE DOCKER Y DOCKER COMPOSE

(Todos los comandos de este manual, marcados con una franja gris, se utilizarán en la consola de comandos)

1) Actualizar los repositorios:

a) Actualizar el índice de paquetes apt:

```
sudo apt-get update
```

b) Instalar los siguientes paquetes para que apt pueda usar un repositorio sobre HTTPS:

```
sudo apt-get install \
   apt-transport-https \
   ca-certificates \
   curl \
   gnupg2 \
   software-properties-common
```

c) Añadir la clave oficial de Docker al administrador de claves de apt (apt-key):

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
```

d) Añadir el siguiente repositorio para poder instalar Docker:

```
sudo add-apt-repository \
   "deb [arch=amd64] \
   https://download.docker.com/linux/debian \
   $(lsb_release -cs) \
   stable"
```

2) Instalar Docker:

a) Actualizar el índice de paquetes apt:

```
sudo apt-get update
```

b) Instalar la última versión de Docker:

```
sudo apt-get install docker-ce
```

c) Comprobar que la instalación se ha realizado correctamente, que dará error si en caso negativo:

```
sudo docker run hello-world
```

3) Instalar Docker Compose:

a) Descargar la última de Docker Compose:

```
sudo curl -L \
https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o \
/usr/local/bin/docker-compose
```

b) Dar permisos de ejecución a la librería donde se encuentra Docker Compose:

```
sudo chmod +x /usr/local/bin/docker-compose
```

c) Comprobar que la instalación se ha hecho correctamente, que devolverá la versión del programa en caso afirmativo:

```
docker-compose --version
```

APÉNDICE II: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE LOGSTASH, ELASTICSEARCH Y KIBANA

(Todos los comandos de este manual, marcados con una franja gris, se utilizarán en la consola de comandos)

1) **Descargar el repositorio "*docker-elk-master*"** (versión 6.2.3) desde el siguiente enlace:

   https://github.com/deviantony/docker-elk/archive/c31545317841dc05bbc9fcf4f19c1528da8a2328.zip.

2) **Descomprimir el interior del archivo .zip descargado en el paso anterior:**

```
unzip /path/docker-elk-c31545317841dc05bbc9fcf4f19c1528da8a2328.zip
```

   - En el comando, *"/path/"* representa la ruta donde se ha descargado el archivo.
   - De aquí en adelante se referirá a la ruta donde se encuentra el archivo descomprimido con "*docker-elk-master*", por lo que se recomienda cambiar el nombre de la carpeta descomprimida a dicho nombre.

3) **Añadir la línea siguiente en el archivo "*docker-compose.yml*", situado en la ruta "*docker-elk-master/*":**

```
./elasticsearch/data:/usr/share/elasticsearch/data
```

   - Esta línea debe añadirse en el apartado *services:elasticsearch:volumes*, dentro del archivo indicado.
   - La adicción de esta línea en el fichero "*docker-compose.yml*" permite la persistencia de los datos almacenados en Elasticsearch.
   - Si se produjese algún error al iniciar el sistema en el paso 5), comprobar que el usuario que está ejecutando el comando indicado en dicho paso tiene suficientes permisos en la carpeta "*docker-elk-master/elasticsearch/data/*".
     En caso de no ser así, modificar los permisos de la carpeta desde la consola de comandos usando *chmod*.

4) **Desplegar el servidor**:

```
docker-compose up
```

   - Este comando debe utilizarse en la ruta "*docker-elk-master/*"

5) Modificar el valor de la variable del sistema *vm.max_map_count* a 262144

```
sysctl -w vm.max_map_count=262144
```

- El valor de la variable "*vm.max_map_count*" es el que utiliza Elasticsearch como referencia para conocer cuanta memoria necesita para almacenar sus índices.
- El valor de esta variable por defecto es 65530, lo que puede dar lugar a errores relacionados con la falta de memoria. Usando el comando anterior se aumenta la cantidad de memoria que utiliza Elasticsearch, retrasando la aparición de ese tipo de errores.

6) Crear el primer *index pattern* (o patrón de índices) en Kibana:

- Este paso se realiza para indicar a Kibana la información de qué índices debe incluir en el *index pattern*.
  Por ejemplo, en el caso actual, los índices de Logstash en Elasticsearch siguen el formato "*logstash-dd-MM-YY*", es decir, se crea un índice por día del año. Entonces, si se crea un *index pattern* llamado "*logstash-**", este contendrá la información de todos los índices de logstash.
- Existen dos opciones para la creación del *index pattern*:
  - Opcion 1: usando la interfaz gráfica de Kibana
    a) Acceder a Kibana mediante el siguiente enlace: http://DIR_IP:5601, donde DIR_IP se corresponde con la IP del equipo.
    b) Acceder a *Management* > *Create index pattern*
    c) Asignarle como nombre "*logstash-**" y dar click a "*Next step*"
    d) En la pestaña "*Configure settings*", seleccionar "*@timestamp*" en el menú desplegable de "*Time Filter field name*"
    e) Finalizar la creación del patrón haciendo clic en "*Create index pattern*".

  - Opción 2: utilizando el siguiente comando:

```
curl -XPOST -D- \
   'http://IP_SERVER:5601/api/saved_objects/index-pattern' \
   -H 'Content-Type: application/json' \
   -H 'kbn-version: 6.2.3' \
   -d '{"attributes":{"title":"logstash-
*","timeFieldName":"@timestamp"}}'
```

APÉNDICE III: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE FILEBEAT. CONEXIÓN ENTRE FILEBEAT Y LOGSTASH

(Todos los comandos de este manual, marcados con una franja gris, se utilizarán en la consola de comandos)

1) **Descargar Filebeat a través de Docker:**

```
docker pull docker.elastic.co/beats/filebeat:6.2.3
```

2) **Cargar el *index template* de Elasticsearch:**

```
docker run docker.elastic.co/beats/filebeat:6.2.3 setup --template \
-E output.logstash.enabled=false -E \
'output.elasticsearch.hosts=["SERVER_IP:9200"]'
```

- En el comando, "*SERVER_IP*" se corresponde con la dirección IP del equipo servidor.
- Este paso se realiza para la configuración de un *index template* en Elasticsearch, es decir, para especificar a Elasticsearch el formato que va a tener la información recibida de Filebeat.

3) **Modificar permisos de los *logs* para que Filebeat pueda acceder a ellos:**

```
chmod 777 /path/to/log1
chmod 777 /path/to/log2
```

- NOTA: este comando se ejecuta para que cualquier usuario tenga acceso a los *logs*, pero evidentemente es una opción poco recomendable. Utilizar otra opción si se conociera otra (en este trabajo esta es la única opción encontrada).

4) **Crear un el archivo de configuración de Filebeat llamado "*filebeat.yml*" con el siguiente contenido** (donde "*SERVER_IP*" se corresponde con la dirección IP del equipo servidor)**:**

```
output.logstash:
  hosts: ["SERVER_IP:5000"]

filebeat.prospectors:
  - type: log
    paths:
      - /path/to/log1
      - /path/to/log2
```

5) **Modificar el archivo de configuración de Logstash *"logstash.conf"* para permitir la conexión con Filebeat:**

- Este archivo se encuentra en el equipo servidor, en la ruta "*docker-elk-master/logstash/pipeline/*".
- En dicho archivo hay que sustituir el conjunto "input { … }" por el siguiente contenido:

```
input {
   beats {
      port => 5000
      client_inactivity_timeout => 86400
   }
}
```

- El parámetro "*port*" indica que va a recibir los *beats* de Filebeat en el puerto 5000, y el parámetro "*client_inactivity_timeout*" indica el tiempo (en segundos) que debe pasar sin recibir un *beat* para que Logstash termine la conexión con un cliente.

6) **Pasar como parámetro a través de Filebeat la IP del equipo cliente actual para poder utilizarlo posteriormente en Kibana** (ejecutar estos comandos en el directorio donde se encuentre el archivo "*filebeat.yml*"):

a) Obtener el alias de la interfaz en uso para la conexión a Internet:

```
interfaceName=$(ip addr show | awk '/inet.*brd/{print $NF; exit}')
```

b) Obtener la IPv4 del equipo cliente actual a través de dicho alias:

```
ip4=$(ip -o -4 addr list $interfaceName | awk '{print $4}' | cut -d/ -f1)
```

c) Añadir la anterior variable de entorno ("*ip4*") a un archivo .env:

```
echo "ip4=$ip4" > .env
```

d) Añadir el parámetro *hostname* en el siguiente paso con el valor de "*ip4*".

7) **Iniciar Filebeat utilizando el archivo de configuración generado en el paso 4):**

- Opción 1: a través del siguiente comando

```
docker run -p 5000:5000 --hostname $ip4 \
-v /path/filebeat.yml:/usr/share/filebeat/filebeat.yml \
-v /path/to/logs:/var/log \
docker.elastic.co/beats/filebeat:6.2.3 --strict.perms=false
```

- Opción 2: creando y utilizando un "*docker-compose.yml*"

  a) Crear un archivo *"docker-compose.yml*", en el directorio donde se encuentra el archivo "*filebeat.yml*", con el siguiente contenido:

```
version: "2"
services:
  filebeat:
    hostname: ${ip4}
    image: docker.elastic.co/beats/filebeat:6.2.3
    command: --strict.perms=false
    ports:
      - "5000:5000"
    volumes:
      -
"/path/to/filebeat.yml:/usr/share/filebeat/filebeat.yml"
      - "/path/to/logs:/var/log"
```

  b) Sustituir, en el fichero anterior, *"/path/to/filebeat.yml*" y *"/path/to/logs*" por la ruta absoluta del archivo "*filebeat.yml*" y de los *logs* que Filebeat va a enviar, respectivamente.

  c) Desplegar el contenedor de Filebeat (utilizar el comando en el directorio donde se haya el archivo "*docker-compose.yml*"):

```
docker-compose up
```

APÉNDICE IV: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE X-PACK

(Todos los comandos de este manual, marcados con una franja gris, se utilizarán en la consola de comandos)

1) Configurar Elasticsearch para instalar y usar X-Pack como *plugin*:

a) Añadir la siguiente línea en el *Dockerfile* de Elasticsearch, situado en el directorio "*docker-elk-master/elasticsearch/*":

```
RUN elasticsearch-plugin install x-pack
```

b) Añadir las siguientes líneas en el archivo "*elasticsearch.yml*" situado en el directorio "*docker-elk-master/elasticsearch/conf/*":

```
xpack.security.enabled: false
xpack.monitoring.enabled: true
```

2) Configurar Kibana para instalar y usar X-Pack como *plugin*:

a) Añadir la siguiente línea en el *Dockerfile* de Kibana, situado en el directorio "*docker-elk-master/kibana/*":

```
RUN kibana-plugin install x-pack
```

b) Añadir las siguientes líneas en el archivo "*kibana.yml*", situado en el directorio "*docker-elk-master/kibana/conf/*":

```
xpack.security.enabled: false
xpack.monitoring.enabled: true
```

3) Configurar Logstash para instalar y usar X-Pack como *plugin*:

a) Añadir la siguiente línea en el *Dockerfile* de Logstash, situado en el directorio "*docker-elk-master/logstash/*":

```
RUN logstash-plugin install x-pack
```

b) Añadir la siguiente línea en el archivo "*logstash.yml*", situado en el directorio "*docker-elk-master/logstash/conf/*":

```
xpack.monitoring.elasticsearch.url: http://SERVER_IP:9200
```

- "*SERVER_IP*" se corresponde con la IP del equipo servidor.

4) Recompilar los contenedores desplegados con el "*docker-compose.yml*":

```
docker-compose up --build
```

APÉNDICE V: MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DE SEARCH GUARD

(Todos los comandos de este manual, marcados con una franja gris, se utilizarán en la consola de comandos)

1) **Generación de certificados TLS/SSL** (mediante el *script* de ejemplo de Search Guard):

   a) Descargar el repositorio del siguiente enlace:

   https://github.com/floragunncom/search-guard-ssl/archive/es-6.0.0.zip

   b) Descomprimir el contenido del archivo descargado: "*search-guard-ssl-es-6.0.0.zip*".

   c) Ejecutar el *script* "*example.sh*" situado en el directorio "*search-guard-ssl-es-6.0.0/example-pki-scripts*". Para ello se accede a dicho directorio y se ejecuta:

   ```
   ./example.sh
   ```

   - Si al ejecutar el *script* diera un error por no tener los permisos suficientes, utilizar el comando *chmod*:

   ```
   chmod 777 ./example.sh
   ```

   d) La anterior ejecución del script creará los certificados necesarios en el directorio anterior ("*search-guard-ssl-es-6.0.0/example-pki-scripts"*). De aquí en adelante en este manual se referirá a esta ruta como "*example-pki-scripts*".

2) **Activar el control de acceso y el cifrado TLS/SSL en la capa de transporte (seguridad base):**

   - En Elasticsearch

     a) Crear una carpeta denominada "*sg*" en el directorio "*docker-elk-master/elasticsearch/config*" donde se almacenarán los certificados necesarios para la configuración de Search Guard.

     b) A la carpeta anterior añadir, de los certificados obtenidos en el paso 1) de este apéndice, los siguientes archivos:
     - "*example-pki-scripts/node-0-keystore.jks*"
     - "*example-pki-scripts/truststore.jks*"
     - "*example-pki-scripts/kirk-keystore.jks*"

c) Crear una carpeta "*bin*" en el directorio "*docker-elk-master/elasticsearch/*". Dentro de este directorio crear un script llamado "*init_sg.sh*" con las siguientes líneas:

```
#!/bin/sh
plugins/search-guard-6/tools/sgadmin.sh \
   -cd plugins/search-guard-6/sgconfig/ \
   -ts config/sg/truststore.jks \
   -ks config/sg/kirk-keystore.jks \
   -nhnv \
   -icl
```

- Este *script* se ejecutará una vez completada la instalación de Search Guard para iniciar este servicio.

d) Añadir las siguientes líneas en el *Dockerfile* de Elasticsearch, situado en el directorio "*docker-elk-master/elasticsearch/*", con el fin de instalar Search Guard y copiar las carpetas creadas en pasos anteriores en el despliegue del contenedor de Elasticsearch:

```
COPY config/sg/ config/sg/
COPY bin/ bin/
RUN elasticsearch-plugin install com.floragunn:search-guard-
6:6.2.3-22.1 \
      && chmod +x plugins/search-guard-6/tools/*.sh \
      && chown -R elasticsearch config/sg/ \
      && chmod -R go= config/sg/
```

e) Añadir las siguientes líneas en el archivo "*elasticsearch.yml*", situado en el directorio "*docker-elk-master/elasticsearch/conf/*", para establecer el cifrado TLS/SSL en el nodo que forma el cluster de Elasticsearch :

```
searchguard.enterprise_modules_enabled: false
searchguard.ssl.transport.keystore_filepath:       sg/node-0-
keystore.jks
searchguard.ssl.transport.keystore_password: changeit
searchguard.ssl.transport.truststore_filepath:
sg/truststore.jks
searchguard.ssl.transport.truststore_password: changeit
searchguard.ssl.transport.enforce_hostname_verification:
false
searchguard.authcz.admin_dn:
        - CN=kirk,OU=client,O=client,L=Test,C=DE
```

- En Kibana:

  a) Añadir la siguiente línea en el *Dockerfile* de Kibana, situado en el directorio "*docker-elk-master/kibana/*", para la instalación del plugin de Search Guard en el contenedor de Kibana:

  ```
  RUN kibana-plugin install
  https://search.maven.org/remotecontent?filepath=com/floragunn
  /search-guard-kibana-plugin/6.2.3-13/search-guard-kibana-
  plugin-6.2.3-13.zip
  ```

  b) Añadir las siguientes líneas en el archivo "*kibana.yml*", situado en el directorio "*docker-elk-master/kibana/conf/*", para que Kibana tenga los permisos suficientes para acceder a los índices de Elasticsearch y extraer la información contenida en ellos:

  ```
  elasticsearch.username: "kibanaserver"
  elasticsearch.password: "kibanaserver"
  ```

  c) Tras realizar estas configuraciones son necesarias unas credenciales usuario-contraseña para acceder a Kibana. Estas credenciales son **admin-admin**.

- En Logstash:

  a) Cambiar, en el archivo "*logstash.conf*", situado en el directorio "*docker-elk-master/logstash/pipeline/*", el conjunto "*output { … }*" por lo siguiente para que Logstash disponga de los permisos suficientes para enviar información a Elasticsearch:

  ```
  output {
     elasticsearch {
        user => logstash
        password => logstash
        hosts => ["elasticsearch:9200"]
     }
  }
  ```

  b) Añadir las siguiente líneas en el archivo "*logstash.yml*", situado en el directorio "*docker-elk-master/logstash/conf/*":

  ```
  xpack.monitoring.elasticsearch.username: admin
  xpack.monitoring.elasticsearch.password: admin
  ```

3) **Recompilar los contenedores desplegados con el "*docker-compose.yml*":**

```
docker-compose up --build
```

4) **Para iniciar SearchGuard,** es necesario ejecutar el siguiente comando una vez se inicie ElasticSearch:

```
docker-compose exec -T elasticsearch bin/init_sg.sh
```

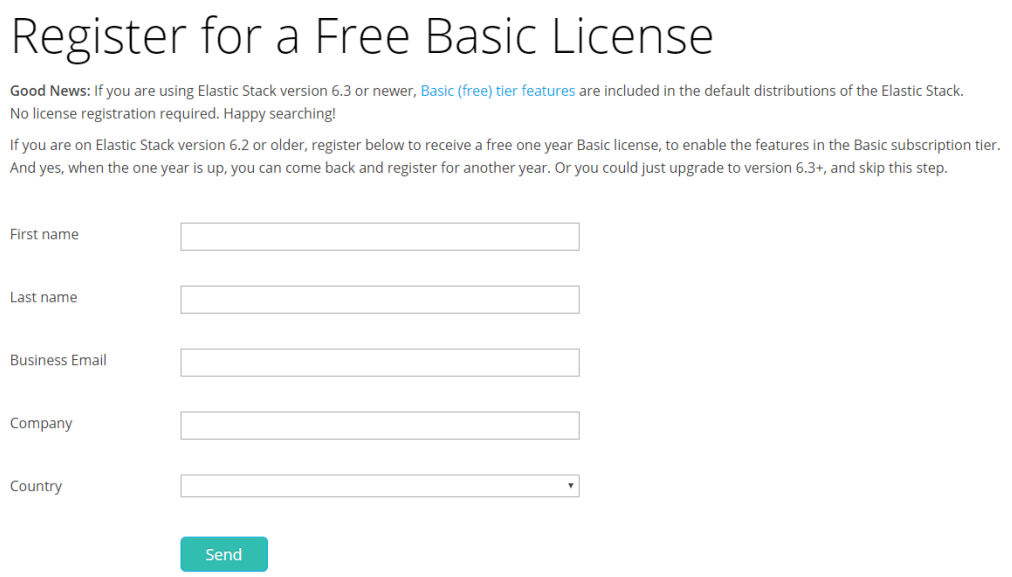
APÉNDICE VI: MANUAL DE OBTENCIÓN E INSTALACIÓN DE UNA LICENCIA TIPO BASIC EN ELASTIC

(Todos los comandos de este manual, marcados con una franja gris, se utilizarán en la consola de comandos)

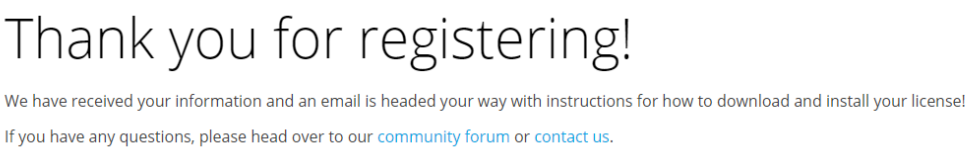
1) **Acceder al siguiente enlace:**

   https://register.elastic.co/xpack_register

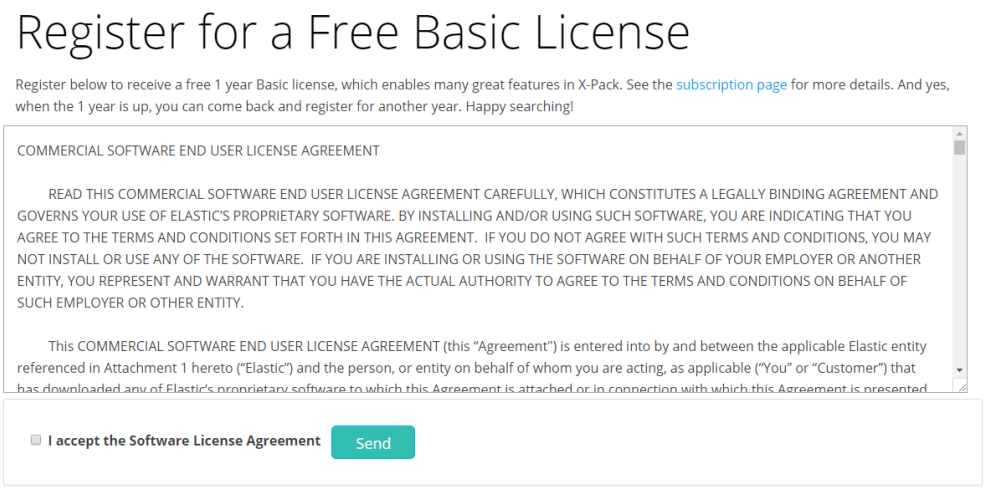
2) **Completar el cuestionario con los datos requeridos y hacer clic en "*Send*".**



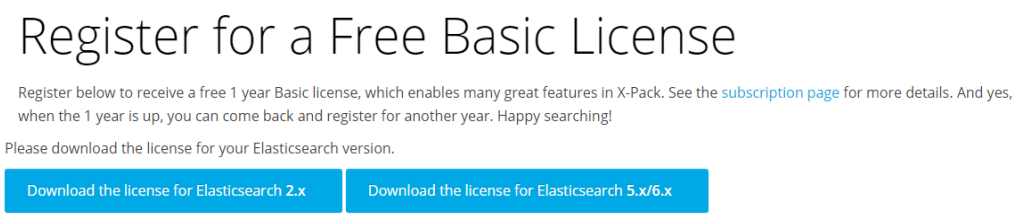
3) **Una vez enviado el cuestionario aparecerá el siguiente mensaje y se recibirá un correo en la dirección especificada en el campo "*Business Email*".**



4) **En el interior del correo se especifica un enlace para descargar la licencia. Dicho enlace lleva a la siguiente pantalla**:



5) **Tras aceptar el acuerdo de licencia de software se accede a esta siguiente página, donde hay que clicar en el botón "*Download the license for Elasticsearch 5.x/6.x*"**. Tras esto se descargará la licencia en formato .json.



6) **Ejecutar el siguiente comando en la consola de comandos para activar la licencia descargada**. El comando debe ejecutarse en el directorio donde esté el archivo .json descargado en el paso anterior.

```
curl -XPUT -u admin 'http://<SERVER_IP>:9200/_xpack/license' -H "Content-Type: application/json" -d @license.json
```

- <*SERVER_IP*>: dirección IP del equipo servidor.
- *license.json*: nombre de la licencia descargada. También puede renombrarse dicha licencia con el nombre indicado ("*license.json*").

BIBLIOGRAFÍA

[1] *Sumo Logic* [consulta: febrero de 2018]. Disponible en: https://www.sumologic.com/.

[2] *Splunk* [consulta: febrero de 2018]. Disponible en: https://www.splunk.com/.

[3] *Hewlett Packard Enterprise. ¿Qué son los contenedores?* [consulta: marzo de 2018]. Disponible en: https://www.hpe.com/es/es/what-is/containers.html.

[4] *1&1 Digital Guide. Docker y otros container: más allá de la virtualización* [consulta: marzo de 2018]. Disponible en: https://www.1and1.es/digitalguide/servidores/know-how/docker-container-las-ventajas-de-los-contenedores-web/.

[5] PERI, N. *Logz.io. Fluentd vs. Logstash: A Comparison of Log Collectos* [consulta: marzo de 2018]. Disponible en: https://logz.io/blog/fluentd-logstash/.

[6] *Apache Solr vs Elasticsearch* [consulta: marzo de 2018]. Disponible en: http://solr-vs-elasticsearch.com/.

[7] YIGAL, A. *Logz.io. Grafana vs. Kibana: The Key Differences to Know* [consulta: marzo de 2018]. Disponible en: https://logz.io/blog/grafana-vs-kibana/.

[8] BERMAN, D. *Logz.io. Filebeat vs. Logstash - The Evolution of a Log Shipper* [consulta: marzo de 2018]. Disponible en: https://logz.io/blog/filebeat-vs-logstash/.

[9] *Docker - Build, Ship, and Run Any App, Anywhere* [consulta: marzo de 2018]. Disponible en: https://www.docker.com/.

[10] *Elasticsearch: RESTful, Distributed Search & Analytics* [consulta: marzo de 2018]. Disponible en: https://www.elastic.co/products/elasticsearch.

[11] *Logstash: Collect, Parse, Transform Logs* [consulta: marzo de 2018]. Disponible en: https://www.elastic.co/products/logstash.

[12] *Kibana: Explore, Visualize, Discover Data* [consulta: marzo de 2018]. Disponible en: https://www.elastic.co/products/kibana.

[13] *Filebeat: Lightweight Log Analysis* [consulta: marzo de 2018]. Disponible en: https://www.elastic.co/products/beats/filebeat.

[14] *X-Pack: Extend Elasticsearch, Kibana & Logstash* [consulta: marzo de 2018]. Disponible en: https://www.elastic.co/products/x-pack.

[15] *Search Guard: Security for Elasticsearch and the ELK Stack* [consulta: marzo de 2018]. Disponible en: https://search-guard.com/.

[16] *Channel Partner: El precio medio de los PC profesionales sube un 12% en un año en Europa* [consulta: abril de 2018]. Disponible en: http://www.channelpartner.es/negocios/noticias/1103413002202/precio-medio-de-pc-profesionales-sube-12-ano-europa.1.html.

[17] MAZUELAS, J. *ELDERECHO.COM: Costes directos y costes indirectos* [consulta: abril de 2018]. Disponible en: http://www.elderecho.com/tribuna/contable/costes_ directos- costes_indirectos_11_685180004.html.